



MATLAB 宝典

- 作者拥有10年MATLAB使用经验，对软件理解深刻
- 全面覆盖数值计算技术，内容深入
- 采用实例的方式讲解，众多实例都取自工程实践
- 对Simulink有详细的介绍，适合相关专业人士阅读

陈 杰 编著



電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn



MATLAB 宝典

MATLAB 是一个非常庞大的数值计算软件，里面有各种模型库，每个库都需要相应的背景知识。但是万变不离其宗，它有很多基础性的、各个库都需要的技术，这是每个MATLAB用户都需要掌握的，本书的目标就是把所有通用功能一网打尽，让每个 MATLAB 用户都可以从中汲取到这些知识。

宝典丛书







责任编辑：张月萍
责任美编：秦　嗬

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书

ISBN 7-121-03378-X　　定价：89.00元

宝 典 丛 书

MATLAB 宝典

陈　杰　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书由浅入深、循序渐进地介绍了MATLAB7.0的知识体系及操作方法。全书共分14章，内容涵盖了MATLAB7.0概述、基础知识、数值运算、数据分析、符号计算、数据的可视化、M语言程序设计、Simulink仿真系统、句柄图形、GUI、文件输入/输出、编译器和应用程序接口等。本书最大的特色在于每一节的例子都是经过精挑细选的，具有很强的针对性，力求让读者通过亲自动手做而掌握基本参数及制作技巧，学习尽可能多的知识。

本书适用于初、中级MATLAB7.0用户，同时也可作为本科生、研究生和教师以及广大科研工作人员学习MATLAB的参考用书。

图书在版编目(CIP)数据

MATLAB宝典 / 陈杰编著．—北京：电子工业出版社，2007.1
（宝典丛书）
ISBN 7-121-03378-X

Ⅰ.M… Ⅱ.陈… Ⅲ.计算机辅助计算－软件包，MATLAB 7.0 Ⅳ.TP391 75

中国版本图书馆CIP数据核字（2006）第129641号

责任编辑：张月萍　　特约编辑：明足群
印　　刷：北京天竺颖华印刷厂
装　　订：三河金马印装有限公司
出版发行：电子工业出版社
　　　　　北京市海淀区万寿路173信箱　　邮编：100036
开　　本：787×1092　1/16　　印张：57　　字数：1623千字
印　　次：2007年1月第1次印刷
定　　价：89.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系电话：（010）68279077，邮购电话：（010）88254888。

质量投诉请发邮件至zlts@phei.com.cn。
服务热线：（010）88258888。

前　言

MATLAB是Mathworks公司推出的一套高性能数值计算和可视化软件，它集数值分析、矩阵运算、信号处理和图形显示于一体，在系统建模和仿真、科学和工程绘图以及应用程序开发等方面有着广泛应用。MATLAB由著名的线性代数软件包LINPAK和特征值计算软件包EISPACK的子程序为基础，发展为一种开发性程序设计软件，因此MATLAB已经由简单的矩阵计算分析软件发展成为通用性极高、带有多种实用工具的运算操作平台。

MATLAB7.0是Mathworks公司近年推出的最新版本，相对于以前的版本，MATLAB7.0有了较大的改进和增补，在开发环境、程序设计、数值处理以及数据可视化方面提供了许多新功能和更为有效的处理方法。

为了帮助众多从业者提高软件使用及操作水平，笔者精心编著了本书。本书依照读者的学习规律，首先介绍基本概念和基本操作，在读者掌握了这些基本概念和基本操作的基础上，再对内容进行深入的讲解，严格遵循由浅入深、循序渐进的原则。本书按照MATLAB7.0内在的联系将各种工具、命令和命令面板交织编排在一起，这样编排虽然不像帮助文档那样有层次感，但是对理解和掌握MATLAB7.0却是大有帮助的。

本书在内容的编排和目录组织上都十分讲究，争取让读者能够快速掌握软件的使用方法。讲解具体知识的时候，尽量避免冗长的知识讲解，直接切入主题，告诉读者如何实现特定功能，让读者在实际操作中熟悉软件的使用。严格制作每一个实例，强调实例效果，这样保证读者的起步层次比较高，在实践的过程中提高制作水平。

和其他书籍相比，本书有以下特点。

1. 内容全面，权威

本书内容全面，它不但表现在介绍某一专项技术时做到全方位的渗透，而且即使本书介绍的仅是MATLAB的某项应用，也会做到对各个技术点逐一攻破。“权威”是指全书的介绍准确，同时还有一定的高度。

2. 取材广泛，内容充实

作者在讲解每一个知识点之前，充分考虑了MATLAB的知识和实践工作的结合，精心挑选数学研究、图形设计、工程运用等各个领域的应用，使读者不仅仅学到MATLAB的操作技巧，而且对创意、思路有所提高。

3. 内容深入

本书的所有实例都有一定的代表性和通用性，并不是为单纯介绍某个命令而选取的，因此有些实例的步骤比较多，综合了MATLAB中的多个知识点，能够提高用户综合使用知识的能力。

4. 讲解仔细

每个实例的制作步骤都以通俗易懂的语言阐述，并穿插讲解和技巧文字，在阅读时就像听课一样详细而贴切。读者只需要按照步骤操作，就可以学习到MATLAB的相关功能。

本书包括的内容

第1章首先带领读者了解MATLAB7.0的应用领域，了解MATLAB7.0的新增功能，熟悉MATLAB7.0的用户界面等入门知识。

第2章介绍MATLAB7.0的基础知识。包括在MATLAB中如何创建数组、操作数组、操作矩阵等基础知识。

第3章主要介绍MATLAB7.0的数值运算，主要包括矩阵分析、线性方程组、矩阵分解、数值积分、函数零点和数理分析等内容。

第4章主要介绍如何使用MATLAB7.0进行数据分析，主要内容包括数据插值、曲线拟合、傅里叶分析、优化和常微分方程等。

第5章主要介绍如何使用MATLAB7.0进行符号计算，主要内容包括符号表达式、符号表达式的运算、符号函数的操作、符号微积分、符号积分变换、符号矩阵计算、符号线性代数方程和符号微分方程等。

第6章主要介绍如何使用MATLAB7.0进行数据可视化，主要内容包括如何绘制二维图形、三维图形以及如何操作三维图形等。

第7章主要介绍如何使用MATLAB7.0进行程序设计，主要内容包括程序设计结构、控制语句、程序的向量化以及如何调试分析程序代码等。

第8章主要介绍Simulink仿真系统的知识，主要内容包括Simulink基础知识、数据类型、基础操作和仿真设置等内容。

第9章主要介绍Simulink仿真系统的高级技术，详细介绍了子系统、封装子系统、使能子系统、触发子系统和S-函数等内容。

第10章主要介绍句柄图形的知识，主要内容包括句柄图形体系、句柄对象操作和高层绘图命令等内容。

第11章主要介绍图形用户界面的制作，主要内容包括如何使用M文件创建GUI、如何使用GUIDE创建GUI、如何添加菜单对象、如何添加控件等。

第12章主要介绍文件的输入和输出，主要内容包括如何在MATLAB中处理外部的二进制文件、文本文件和图像等。

第13章主要介绍MATLAB7.0中的编译器，主要内容包括编译器的安装、设置和对应的命令，以及如何创建独立应用程序等。

第14章主要介绍应用程序接口，主要内容包括如何创建MEX文件、MAT文件，以及如何使用MATLAB7.0的引擎技术等。

本书具有知识全面、实例精彩、指导性强的特点，力求以全面的知识性及丰富的实例来指导读者透彻学习MATLAB7.0各方面的技术。本书适用于初、中级MATLAB7.0用户，同时也适合使用MATLAB的本科生、研究生和教师以及广大科研工作人员作为参考用书，对高级读者也有一定的启发意义。

本书中的MATLAB文件请读者访问http://www.hxex.cn的“资源下载”栏目查找并下载。

作　者

2006年12月

目　录

第1章　MATLAB 概述

本章包括

- MATLAB7.0 的安装
- MATLAB7.0 的工作环境
- MATLAB7.0 的常见命令
- MATLAB7.0 的帮助系统

软件名称 MATLAB 是由 MATRIX 和 LABORATORY 两个英文单词的前 3 个字母组合而成的。最初版本的 MATLAB 出现在 20 世纪 70 年代，用 FORTRAN 语言编写，主要功能是实现程序库的接口功能。在 20 世纪 90 年代，MATLAB 发展成为国际公认的标准计算软件，在数值计算方面功能十分强大，从这个时候起，MATLAB 的内核采用 C 语言编写，增加了数据视图功能。在 MATLAB 推向市场之后，由于软件良好的开放性和运行的可靠性，淘汰了该行业其他各种软件，许多工作开始在 MATLAB 平台上重建。

1.1　MATLAB7.0 简介

MATLAB 从第一个版本到目前发布的第 14 个版本 MATLAB7.0（Release 14），软件本身有了很大的改善，增加了很多新功能和有效的操作方法。下面简要对这些内容进行介绍。

- **开发环境**：在 MATALB7.0 中，用户可以同时使用多个文件和图形窗口，用户可以根据自己的习惯和喜好来定制桌面环境，同时还可以为自己常用的定义快捷键。
- **代码开发**：支持函数嵌套、有条件中断点，可以使用匿名函数定义单行函数。
- **数值处理**：在最新的版本中，单精度算法、线性代数可以方便用户处理更大的单精度数据，ODE 可以求解泛函数，操作隐式差分等式和求解多项式边界值问题。
- **数据可视化**：提供新的绘图界面窗口，用户可以不输入 M 函数代码而直接在界面窗口中交互性地创建并编辑图形，同时可以直接从图形窗口中创建对应的 M 代码文件。
- **文件 I/O 和外部应用程序接口**：支持读入更大的文本文件，支持压缩格式的 MAT 文件，用户可以动态加载、删除或者重载 Java 类，支持 COM 用户接口等。

1.2　MATLAB7.0 的安装

MATLAB 是一个功能强大的数学工具软件，只有在适当的系统环境中才能正常运行。相对于 MATLAB7.0 之前的几个版本，MATLAB7.0 在安装时给用户提供了更为个性化的条件。本节将介绍 MATLAB7.0 在 PC 机的 Windows 操作系统中的典型安装方法。



在本节中只介绍 MATLAB7.0 在 Windows 操作系统中的安装方法，但是，MATLAB7.0 本身适合于许多机型和操作系统，例如 Macintosh 和 UNIX 工作站等。在本书的后面章节中，如果没有特别说明，所有的操作都是在 Windows XP 操作系统下的 MATLAB7.0 中进行的。

在一般情况下，当用户将 MATLAB7.0 的安装光盘插入光驱后，会自动启动"安装向导"。如果向

导没有自动启动，可以打开安装光盘中的setup.exe应用程序，启动安装向导。

在安装过程中出现的所有界面都是标准界面，用户只需要按照界面中的提示进行操作，输入用户名、单位名以及软件产品的序列号等。由于MATLAB7.0的安装界面相对于MATLAB之前的版本有了较大改变，下面主要介绍几个明显改变的界面。

首先，用户需要在安装界面中选择"Custom"选项才能自行选择软件安装的组件和目录等，如图1.1所示。

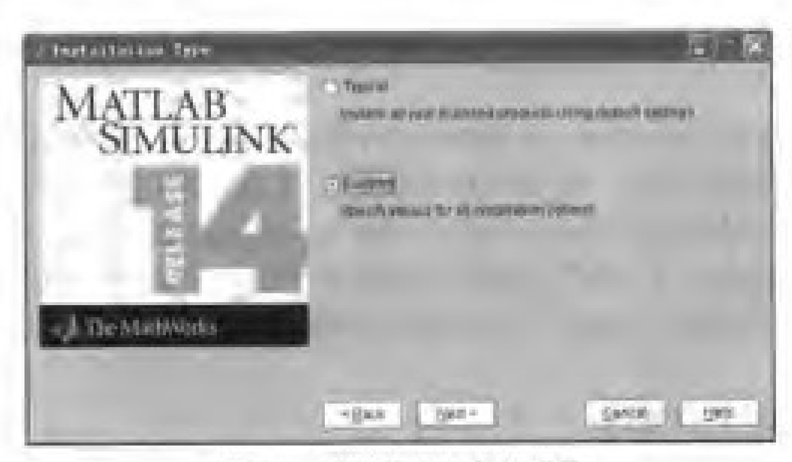


图1.1　选择自定义安装选项

当用户选择"Custom"选项后，单击界面中的"Next"按钮，就会出现MATLAB安装选项的界面，用户可以在该界面中选择需要安装的组件，如图1.2所示。

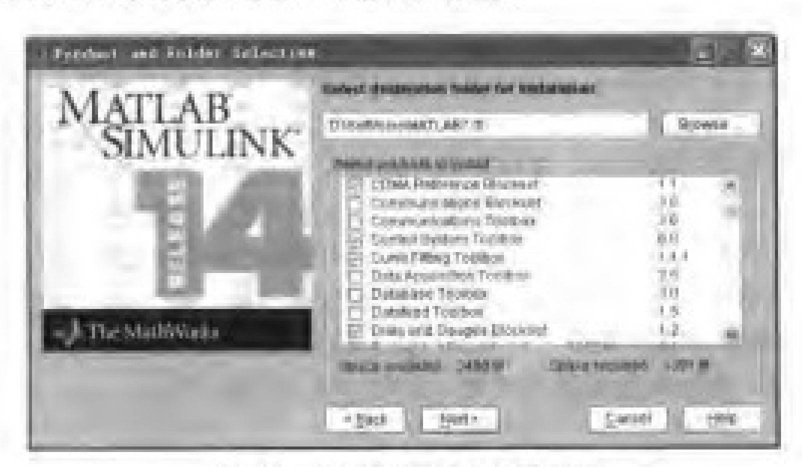


图1.2　MATLAB组件选择界面

在MATLAB组件选择界面的上面选框中，用户需要选择MATLAB软件的安装位置和软件的名称。用户可以使用任意名称，同时也可以将MATLAB软件安装在硬盘中的任何位置。用户可以单击该界面右上方的"Browse"按钮，选择相应的安装路径，也可以直接在选框中输入安装路径名称。



在默认情况下，系统会将MATLAB安装在系统盘中，用户可以使用该默认安装路径。由于MATLAB本身所占空间和资源较大，同时，如果用户经常使用MATLAB软件，会产生很多的临时文件，这将给系统盘增加负担。因此，建议用户将MATLAB安装在其他磁盘中。

在MATLAB组件选择界面的下面选框中，用户需要选择MATLAB软件的安装组件。用户可以直接

勾选相应的组件，进行安装。默认情况下，系统选中所有的组件，但是，对于一般的用户而言，很多组件的专业性过强，没有安装的必要。所以，用户应该根据自己的需要选择安装的组件。

为了帮助用户了解组件的构成，下面给出比较典型的组件组合方式，如表 1.1 所示。

表 1.1 MATLAB 的安装组件

组件名称	说明
MATLAB	必要组件，用户需要选择该组件，否则无法安装其他部分。该组件是整个软件的核心部分，为整个软件系统提供 MATLAB 工作环境
Simulink Symbolic Math Optimization Matlab Complier	MATLAB的通用性组件，可以完成MATLAB的常见功能。建议用户安装这些组件
Control System Curve Fitting Statistics ……	MATLAB 的常用专业性组件，建议用户根据需要选择组件



一般来讲，随着用户对MATLAB使用的深入，可能随时需要安装新的工具箱。如果这个工具箱在安装光盘中已经包括，用户可以再次使用光盘进行安装。如果该工具箱是新增的，用户可以访问MathWorks公司的官方网站，单独下载工具箱的安装文件。

当用户选择MATLAB的安装组件后，单击组件选择界面中的 "Next" 按钮，就会出现使用选项的界面，用户可以在界面中选择 MATLAB 的使用选项，如图 1.3 所示。

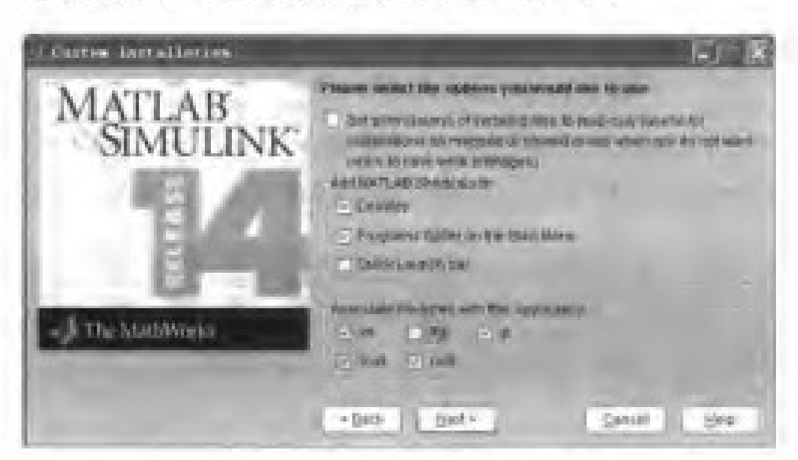


图 1.3 MATLAB 选项界面

在MATLAB选项界面的上部，MATLAB为用户设置软件的密码。如果用户选中该选项，则可以设置操作密码。其他用户只能读取 MATLAB 的所有相关文件，而不能编辑其中任何文件。当用户使用的操作系统是多人共用时，为了保护自行设置和编辑的 MATLAB 文件，建议用户选中该选项。

在该界面的中部，用户可以选择为 MATLAB添加快捷方式的位置。系统提供桌面、开始菜单和快速启动栏三个位置。用户可以根据需要选择添加的位置，一般建议用户选中 "桌面" 和 "开始菜单" 选项。这样就可以很方便地启动 MATLAB。

在该界面的底部，用户可以选择和MATLAB关联的文件扩展名。当用户选中相应的文件扩展名后，在默认的情况下，系统使用MATLAB打开这些扩展名的文件。

1.3 MATLAB7.0 的工作环境

在上节中，已经将MATLAB7.0安装到相应的硬盘上，在本节中，用户可以启动MATLAB，查看MATLAB7.0的工作环境。

在一般情况下，可以使用两种方法来启动MATLAB7.0。在上节中，将快捷方式添加在桌面上，因此可以双击桌面上的快捷方式图标，打开如图1.4所示的操作界面（Desktop）。

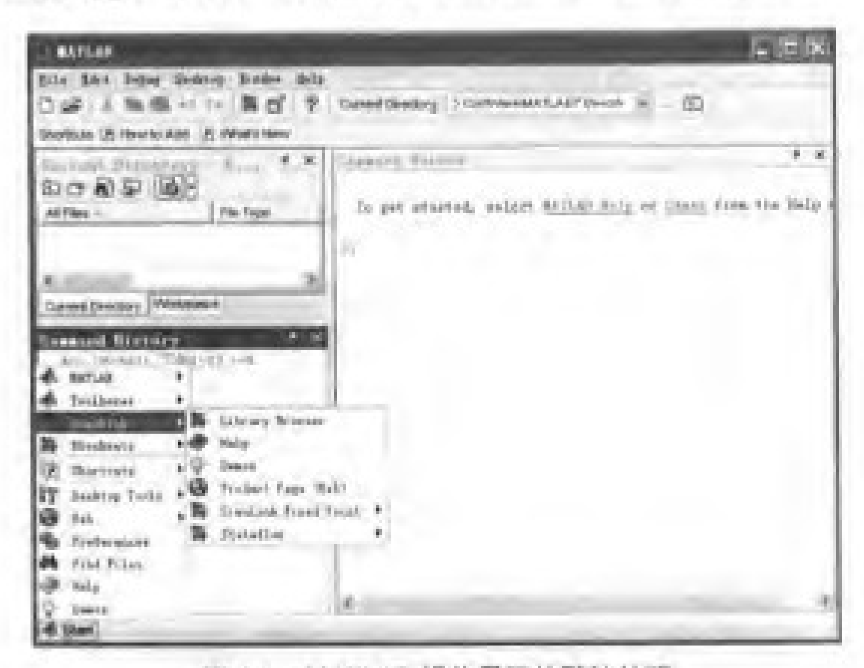


图 1.4　MATLAB 操作界面的默认外观

如果用户没有添加MATLAB的桌面快捷方式，则需要使用电脑选择路径matlab7.0\bin\win32文件夹中的MATLAB.exe应用程序，同样可以打开MATLAB的操作界面。这两种方法的结果是完全相同的。



尽管用户使用上面两个方法都可以启动MATLAB，但是，两个方法还是有区别的。如果用户使用第二种方法启动MATLAB，系统会将matlab7.0\bin\win32作为一个默认的目录。因此，建议用户使用第一种方法来启动MATLAB。

1.3.1 操作界面简介

MATLAB7.0延续了MATLAB6.x版本的操作界面。该操作界面中包含大量的交互性工作界面，例如通用操作界面、工具包专用界面、帮助界面和演示界面等。这些交互性界面组合在一起，构成MATLAB的默认操作界面。

在默认情况下，MATLAB的操作界面包含指令窗、历史指令窗和工作空间浏览器3个最常见的界面，同时，在窗口的左下角为“开始”按钮。



在默认情况下，用户还可以在MATLAB的操作界面中看到名称为"Current Directory"的交互界面选项卡。如果用户单击该选项卡，该交互界面就会出现。该交互界面是一个十分常用的界面——当前目录窗。

下面详细介绍MATLAB中常见的几个交互界面。

- **命令窗口（Command Window）**：这是MATLAB操作界面中最为重要的窗口，也是用户进行各种操作的主要窗口。在这个窗口中，用户可以输入各种MATLAB的命令、函数和表达式。同时，所有操作和运算的结果也会在该窗口中出现（图形结果会单独显示）。
- **历史命令窗口（Command History）**：在默认情况下，该命令窗口出现在MATLAB操作界面的左下方。这个窗口中记录用户已经操作过的各种命令、函数和表达式。之所以记录这些信息，主要功能有两个：方便用户回忆之前的操作，也可以方便用户对这些历史信息进行编辑（例如：复制、重运行等）。
- **当前目录窗口（Current Directory）**：在默认情况下，该命令窗口出现在MATLAB操作界面的左上方的后台。在这个窗口中，用户可以设置当前目录，展示目录中的M文件或者MAT文件等，同时，用户可以编辑M文件等。
- **工作空间浏览器（Workspace Browser）**：在默认情况下，该命令窗口出现在MATLAB操作界面的左上方的前台。在这个窗口中，用户可以出查看工作空间中所有变量的类别、名称和大小。用户可以在这个窗口中观察、编辑和提取这些变量。
- **开始按钮（Start）**：这个按钮是MATLAB6.5版本后增加的按钮。单击这个按钮以后，会出现MATLAB的现场菜单。这个菜单中分为两个部分，上半部分的菜单包含各种交互界面，下半部分的菜单选项的主要功能是窗口设置、访问MATLAB公司的网页、查看帮助文件等。

1.3.2 运行命令窗口（Command Window）

在本节的前面部分已经介绍过，命令窗口是MATLAB的主要操作界面。关于MATLAB的大部分操作命令和结果都需要在命令窗口中进行。

由于命令窗口在默认情况下位于MATLAB操作界面的右方，用户可以单击命令窗口右上角的按钮，使命令窗口脱离操作界面，得到的命令窗口如图1.5所示。



图1.5　单独的命令窗口

和MATLAB之前的版本相比，MATLAB7.0命令窗口的菜单栏有了一些调整。例如，MATLAB7.0的菜单栏中相对于MATLAB6.5，增加了"Debug"和"Desktop"选项，减少了"View"选项。其中，增加"Debug"选项是为了方便用户编译M文件，"Desktop"选项则为方便用户在界面重放方式上调换。

从MATLAB6.x版本以后，当MATLAB在Windows操作系统中运行时，命令窗口都会出现命令行提

示符“>>”。



如果用户希望将命令窗口回放到MATLAB的操作界面中，可以选择命令窗口中的“Desktop” ⇨ “Dock Command Window”命令，也可以直接单击菜单栏中的按钮。

1.3.3 命令窗口的显示方式

在前面小节中，用户已经基本了解了命令窗口的外观，在本小节中将主要介绍控制命令窗口的命令和操作设置，希望用户全面了解MATLAB的命令系统。

在默认情况下，MATLAB7.0对命令窗口中的字符或者数码设置不同的颜色，这样会使用户方便地查看各种信息。用户可以根据自己的需要，对命令窗口的字体风格、大小和颜色等进行自定义的设置。

选择MATLAB的操作界面或者命令窗口中的“Files” ⇨ “Preferences”命令，打开“Preferences”对话框，用户可以在其中设置字体格式等，如图1.6所示。



图1.6 参数设置对话框

选择参数设置对话框左栏中“Fonts”选项，在对话框中的右侧会显示命令窗口的字体属性。在默认情况下，MATLAB将命令窗口（Command Window）、历史窗口（Command History）和M文件编辑器（Editor）中的字体设置为相同：字体类型是Monospace，字体属性是Plain，大小为10。而将帮助导航（Help Navigator）、当前记录窗口（Current Directory）、HTML文本文字、工作空间浏览器（Workspace）和内存数组编辑器（Array Editor）中的字体设置为相同：字体类型是SansSerif，字体属性是Plain，大小为10。

对于上面两种字体类型，用户都可以在对应选项的下拉菜单中选择新的属性，然后单击参数设置对话框中的“OK”按钮，完成属性的设置。



当用户修改字体的属性之后，这些设置都将被永久保留。这些设置不会随着MATLAB的关闭或者开启而改变。但是，用户只能改变各个局部的字体属性，不能改变字体的分布设置。也就是说，用户可以修改命令窗口的字体属性，但是命令窗口的字体和历史窗口的字体属性永远相同。

和设置字体属性类似，用户可以为不同类型的变量设置颜色，以示区别。选择参数设置对话框中左栏的"Colors"选项，在对话框中的右侧设置系统的字体颜色，如图 1.7 所示。



图 1.7 设置字体颜色

在上面的参数设置对话框的上部，用户可以设置MATLAB操作界面的工具字体颜色。默认情况下，MATLAB 会使用系统字体的颜色，用户可以根据需要取消选中"Use system colors"选项，然后选择字体颜色。

在对话框的中部，用户可以为各种类型的字符设置颜色。MATLAB 提供 6 种不同类型的字符：关键字、字符串、系统命令、注释、未结束的字符串和错误提示等。用户可以根据自己的喜好来修改这些不同类型的字符颜色。

在对话框的下部，显示的是用户设置的字体样式。用户可以根据这个结果及时调整字体的颜色。如果用户对自行修改的字体颜色不满意，可以单击"Restore Default Colors"按钮，重新启用系统默认的字体颜色。



尽管MATLAB 为用户提供十分宽松的环境来设置不同的颜色，但是，用户不能将不同类型的字符串设置为相同的颜色。例如，用户可以修改"关键字"的颜色，但是不能将"关键字"和"注释"设置成相同的颜色。

1.3.4 数值结果的显示方式

在默认情况下，命令窗口的数值结果都是以 format short g 的格式来显示的，用户可以根据计算的要求自行设置数据结果的显示方式。



上面描述的是 MATLAB 数值结果的显示方式，而不是数值的计算精度。MATLAB 只是为了显示的简洁才采用较少位数的显示，在实际存储和计算过程中使用的都是双精度型。

表 1.2 显示的就是 MATLAB 中数值显示格式的种类、命令和含义。

表 1.2　数据显示方式的常见命令

命令	说明	举例
format format short	通常显示数值的小数有效位数是 4，最多也不会超过 7 大于 1000 的数值，用 5 位有效数字的科学记数形式来显示	256.375 会被显示为 256.3750 2563.75 被显示为 2.5638e+003
format long	15 位数字表示	2.56375332457890
format short e	5 位科学记数表示	2.5638e+00
format long e	15 位科学记数表示	2.56375332457890e+00
format short g	从 format short 和 format short e 中选择最佳的记数方式	2.5638
format long g	从 format long 和 format long e 中选择最佳的记数方式	2.563753324578901
format rat	用近似有理数表示	3579/1396
format hex	用十六进制数表示	400482911a609f08
format bank	使用金融数据	2.56

根据上面表格的介绍，用户可以直接在 MATLAB 的操作界面中输入相应的命令，查看同一个变量 a 的不同显示结果，如图 1.8 所示。

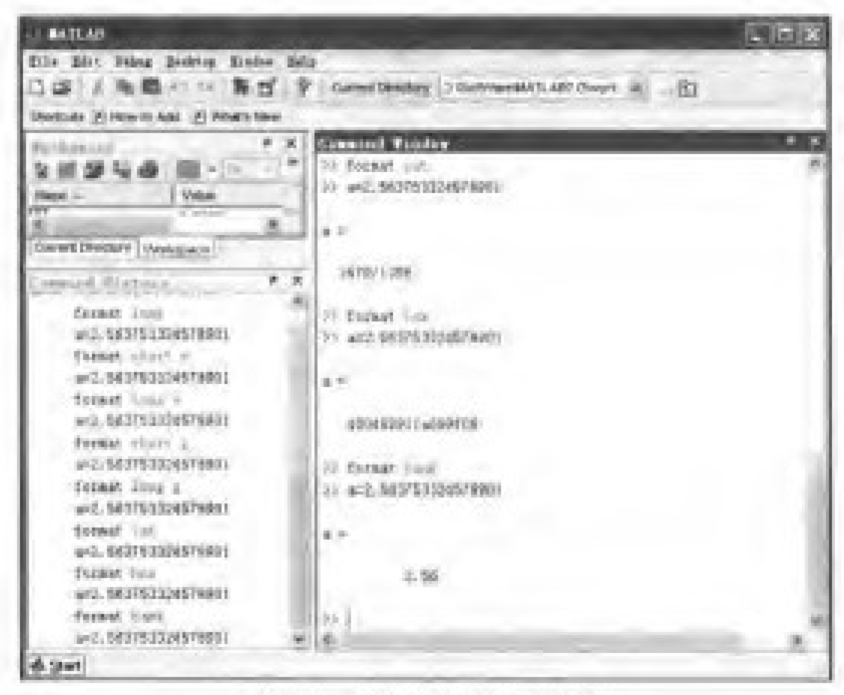


图 1.8　设置数值的显示格式

可以看出，对于同一个变量 a=2.563753324578901，用户使用不同的格式命令会在 MATLAB 中显示出不同的结果。



从上面的操作结果可以看出，对于数值格式的设置，如果使用的是格式命令，相应的设置只对当前的 MATLAB 命令窗口有效，一旦 MATLAB 窗口关闭，这些设置也会失效，系统会恢复数值的默认格式。

1.3.5 命令窗口的标点符号

在MATLAB命令窗口中，不同的标点符号具有不同的运算含义，用户有必要了解各种标点符号的具体含义，如果灵活使用标点符号，会给运算带来很大的方便。表1.3归纳了各种常见标点的作用。

表1.3 MATLAB常见标点的功能

名称	标点	作用
空格		输入变量之间的间隔；数组元素的分隔符
分号	;	用作命令的结束表示，同时不显示结果；数组元素的行间分隔符
冒号	:	用来生成一维数值数组
逗号	,	输入变量之间的间隔；数组元素的分隔符
黑点	.	数值中的小数点
注释号	%	用在数据行的开头，表示该数据行是非执行的注释行
方括号	[ ]	输入数组的时候用
续行号	……	由三个或者三个以上的黑点组成，标号使下行是该行的继续，构成整体

下面列举一些简单的例子，介绍常用标点的功能。

例1.1 在MATLAB中输入矩阵。

具体的输入步骤如下：

step 1 在MATLAB的命令窗口中输入下列内容：

```
A=[ 2,4,6;3,5,7;8,9,10]
```

step 2 按"Enter"键，结束输入并执行命令，得到的结果如图1.9所示。

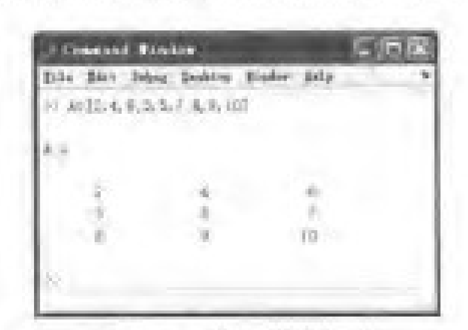


图1.9 输入数值矩阵



从上面的操作结果可以看出，数值矩阵可以直接输入方括号"[ ]"。而矩阵中的数值间隔使用的是逗号，每个数据行之间的分隔使用的是分号。最后，所有的标点都必须在英文状态下输入，MATLAB无法辨识中文的标点符号。

例1.2 在MATLAB命令窗口中输入下面的续行命令：

```
B=1 + 25 - 36 +.....
37 - 58 + 77
```

按"Enter"键，结束输入并执行命令，得到的结果如图1.10所示。



图 1.10　续行输入命令行



上面的案例是在MATLAB的命令窗口环境中进行的，如果用户是在Notebook运行上面的命令，则不能使用续行号，只能让其自动换行。

例 1.3　在 MATLAB 中进行数组点乘。

具体的输入步骤：

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> % 例 1.3.3-3 演示标点符号的用法
>> C=[1,2,3].*[4,5,6]
```

step 2 按"Enter"键，结束输入并执行命令，得到的结果如图 1.11 所示。



图 1.11　MATLAB 中的点乘

上面的计算结果是两个数组中对应元素的乘积，例如，4=1*4；10=2*5；18=3*6。



在上面的操作界面中，首先使用注释号 % 引出了注释行，在下面的计算行中首先输入黑点，然后输入乘号，得到点乘的结果。如果在上面的输入过程中，没有输入黑点，MATLAB 会提示计算错误，因为这两个矩阵无法相乘。

1.3.6　输入变量

在 MATLAB 的计算和编程过程中，变量和表达式都是最基础的元素。如果用户需要深入学习 MATLAB，十分有必要了解 MATLAB 关于定义变量和表达式的基础规定。

在 MATLAB 中，为变量定义名称需要满足下列规则。

- 变量名称和函数名称有大小写区别。对于变量名称 NumVar 和 numvar，MATLAB 会认为是不同的变量。exp 是 MATLAB 内置的指数函数名称，因此，如果用户输入 exp(0)，系统会得出结

果 1；而如果用户输入 EXP(0)，MATLAB 会显示错误的提示信息 "??? Undefined command/function 'EXP'"，表明 MATLAB 无法识别 EXP 的函数名称，如图 1.12 所示。



图 1.12 函数名称区别大小写

- 变量名称的第一个字符必须是英文字符。在 MATLAB6.5 以后的版本中，变量名称最多可以包含 63 个字符。因此，变量 5Var、_matrix 等都是不合法的变量名称。
- 变量名称中不可以包含空格或者标点符号，但是可以包括下划线。因此，变量名称 Num_Var 是合法的，但是变量名称 Num、Var 则是不合法的。



尽管MATLAB对于变量名称的限制较少，笔者还是建议用户在设置变量名称时考虑变量的含义。例如，在 M 文件中，变量名称 inputname 就比名称 a 易于理解。

在上面的变量名称规则中，没有限制用户使用 MATLAB 的预定义变量名称，但是根据笔者经验，还是建议用户不要使用 MATLAB 预先定义的变量名称。因为，用户每次启动 MATLAB，系统就会自动产生这些变量，表 1.4 列出了常见的预定义变量名称。

表 1.4 MATLAB 中的预定义变量

预定义变量	含义
ans	计算结果的默认名称
eps	计算机的零阈值
inf(Inf)	无穷大
pi	圆周率
NaN(nan)	表示结果或者变量不是数值

MATLAB没有限制用户使用上面这些预定义变量，用户可以在MATLAB的任何文件中将这些预定义变量重新定义，赋予新值，然后重新计算。下面的例子可以说明用户使用MATLAB内置的预定义变量。

例 1.4 下面的例子说明如何在 MATLAB 中使用预定义变量，如图 1.13 所示。

上面的详细程序清单如下：

```
>> %演示用户重新定义预定义变量
>> pi                          %显示系统的预定义变量 pi

ans =

    3.1416
>> R=6;                        %定义半径
>> perimeter=2*pi*R            %计算周长

perimeter =
```

```
      37.699
>> pi=3.50;                    %重新定义变量pi
>> perimeter=2*pi*R            %重新计算周长

perimeter =

    42
>> clear;                      %清除用户定义的变量pi和R
>> R=6;                        %定义半径
>> perimeter=2*pi*R            %重新计算周长

perimeter =

      37.699
```



图 1.13　在 MATLAB 中使用预定义变量

在上面的程序清单的第一行中，用户直接输入了"pi="，可以显示系统的预定义变量 pi，在默认情况下，MATLAB 会以"format short g"的数值格式显示系统预先定义的数值 3.1416。在后续的程序中，用户定义了变量 R，然后计算变量周长 perimeter，得到的结果是 37.699。

在后面的步骤中，用户重新定义变量 pi，将其数值设置为 3.50，然后重新计算周长，得到的结果是 42，这就表明 MATLAB 已经将变量 pi 值改为 3.50。也就是说，MATLAB 接受用户重新定义的预定义变量 pi 的数值。

然后使用命令 clear，清除前面步骤中定义的所有变量，重新定义半径，然后再次计算周长，得到的结果是 37.699。这就表明，MATLAB 又将变量 pi 的数值重新设置为 3.1416。



在 MATLAB 的 IEEE 算法规则中，被 0 除是允许的，这不会导致任何程序的中断，只是给出警告信息然后使用名称 Inf 或者 NaN 来记述，这些名称可以在后面步骤得到合理运用。

1.3.7 处理复数

在一般的数学运算软件中，复数属于一般的变量。之所以在本小节中单独介绍关于复数的问题，是因为在 MATLAB 中将复数作为一个整体处理，而不是像其他程序语言那样把实部和虚部分开处理。而复数的虚数单位用预定义变量 i 或者 j 表示。

在 MATLAB 中，核心处理工具是矩阵，因此用户需要了解复数矩阵的处理方法和普通复数单数的差别。下面利用一些简单实例来说明 MATLAB 如何处理复数。

例 1.5 在 MATLAB 中输入复数 $z_1=8+10i$, $z_2=12+6i$, $z_3=25e^{\frac{\pi}{3}i}$。

具体的输入步骤如下：

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> %显示如何在 MATLAB 中输入复数
>> z1=8+10i;                    %直接按照直角坐标的方式输入
>> z2=12+6*i;                   %运算符构成的直角坐标的方式输入
>> z3=25*exp(i*pi/3);           %运算符构成的极坐标的方式输入
>> A=[z1,z2,z3]
```

step 2 按“Enter”键，结束输入并执行命令，得到的结果如图 1.14 所示。

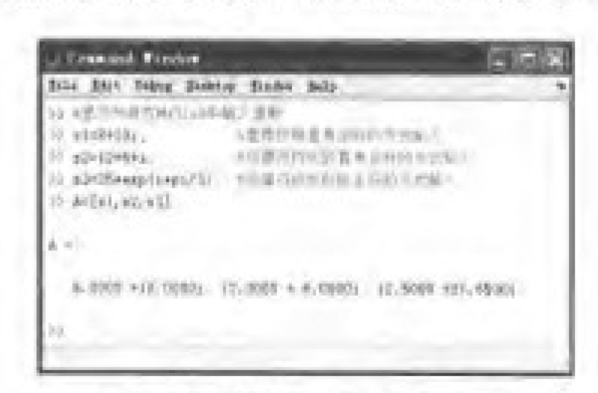


图 1.14 在 MATLAB 中输入复数

在上面的操作界面中，复数 z1 的输入方式是直接按照书面习惯完成的，在这种书写格式中 10i 是一个完整的虚数部分，因此在 10 和 i 之间不允许有任何的空格。这种书写格式符合大家的习惯，但是仅限于使用在复数标量中，不能使用在后面的复数矩阵中。

复数 z2 的输入方式则将虚数部分分开成 6 和 i，同时使用乘号连接。这种方式适用复数矩阵中。需要提醒用户的是，如果可以使用第一种书写格式，尽量使用第一种书写格式，因为这种格式比复数 z2 的输入方式运算速度要快。对于大型的复数矩阵，这种运算速度差别十分明显。

例 1.6 在 MATLAB 中输入复数矩阵并进行矩阵运算。

具体的输入步骤如下：

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> %显示如何使用复数矩阵
>> A=[1,3,5;7,9,11]-[2,4,6;8,10,12]*i;                  %使用数组输入复数矩阵
>> B=[1+2*i,3+4*i;5+6*i,7+8*i;9+10*i,11+12*i];          %使用元素输入复数矩阵
>> C=A*B
```

step 2 按"Enter"键，结束输入并执行命令，得到的结果如图1.15所示。

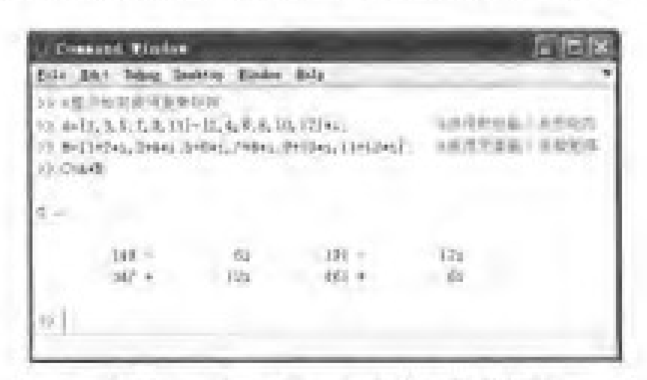


图1.15　在MATLAB中输入复数矩阵

在上面的例子中，使用数组方式输入了复数矩阵A，而使用了数组元素方式输入复数矩阵B，这两种方式都是十分常见的输入方式，效果相同，用户可以根据习惯选择。从输入的繁易程度来看，第一种方法更加简单，建议用户使用这种方法输入复数矩阵。



复数矩阵的乘法和实数矩阵的乘法的计算方法是完全相同的。在上面的实例中，复数矩阵A的维度是23，而复数矩阵B的维度是32，因此两个矩阵相乘得到的结果是一个22矩阵。

例1.7　在MATLAB中计算上例中复数矩阵C的实部、虚部、模和相角。

具体的输入步骤：

step 1 在MATLAB的命令窗口中输入下列内容：

```
>>  %处理复数参量（实部、虚部、模和相角）
Real=real(C);    Imag=imag(C);              %计算复数的实部、虚部
Mag=abs(C);      Phase=angle(C)*180/pi;     %计算复数的模、相角
```

step 2 依次输入上面的各个变量名称，按"Enter"键，结束输入并执行命令，得到的结果如图1.16所示。



图1.16　在MATLAB中计算复数的参量



上面实例中的函数real、imag、abs、angle等都是MATLAB内置的函数，用来处理和计算复数的参量。同时，这些函数的参数都可以是数组类型，这样函数可以对数组元素逐个发生作用。

1.3.8 命令窗口的控制命令

在MATLAB的命令窗口中进行各种操作的时候，用户会经常遇到一些编辑工作，例如：清除窗口的程序语句，清除图形或者关闭MATLAB程序等。这些操作在MATLAB7.0中大部分都可以使用对应的菜单或者功能按钮来实现，但是，在用户编写M文件的时候，还是有必要使用这些控制命令，因此，在本小节中归纳了关于MATLAB常见的控制命令和对应功能，如表1.5所示。

表 1.5 MATLAB 中的常见控制命令

命令	功能
clf	清除图形窗
clc	清除命令窗口中的显示内容
type	显示指定M文件的内容
clear	清除MATLAB工作空间中保存的变量
exit/quit	退出MATLAB程序

上面这些控制命令在整个MATLAB程序中都是通用的，也就是说，既可以在命令窗口中输入这些控制命令，也可以在M文件或者MAT文件的程序语句中使用这些控制命令，功能是完全相同的。例如，在命令窗口中输入clear命令，将会清除工作空间中的变量；如果在M文件中输入clear命令，也会清除变量。



上面这些控制命令一直都是MATLAB中的基本通用命令，在各个版本都适用。如果用户使用的是MATLAB6.x或者更低版本，都可以使用上述命令。

1.3.9 使用历史窗口

在前面已经基础介绍了历史窗口（Command History）的作用，在本小节中，将详细介绍如何合理使用MATLAB的历史窗口的功能。在MATLAB中，利用历史窗口可以验证用户即时的想法，相当于随时进行各种尝试和试验，边做边想。同时，MATLAB还提供一个有效工具实录命令diary。下面详细介绍。

在默认的情况下，历史窗口位于MATLAB操作桌面的左下方的前台，用户可以单击历史窗口右上方的按钮，查看独立的历史窗口，如图1.17所示。



图 1.17 历史窗口

历史窗口的菜单栏和命令窗口的菜单栏相同。如果用户希望将历史窗口嵌回到MATLAB的操作界面中，可以选择历史窗口中的 "Desktop" ⇨ "Dock Command Window" 命令，也可以直接单击菜单栏中的按钮。

在历史窗口中，记录着用户在MATLAB命令窗口中输入的所有命令行（除非用户人为地删除历史窗口中的记录）。一般而言，完整的历史记录包括：用户每次启动MATLAB的时间，每次启动MATLAB的所有命令行。

用户不仅能在历史窗口中查看命令窗口中运行过的所命令行，而且可以根据需要编辑这些命令行。下面列举几个常见的编辑功能。

- **复制命令行**：这种编辑功能适用于使用原来部分的命令行。例如，用户需要输入新的命令行，有部分命令行和历史命令行重复，则可以在历史窗口中点亮相应的命令行，然后单击鼠标右键，在弹出的快捷菜单中选择 "Copy" 选项，如图 1.18 所示。



图 1.18　复制历史命令行

复制历史命令行后，可以在命令窗口中的任何地方粘贴这些命令行，如图 1.19 所示。



图 1.19　粘贴历史命令行

粘贴历史命令行后，用户可以在此之后继续输入新的命令行，这样就节省了用户重新输入这些命令行的时间。



由于上面的操作需要在历史窗口和命令窗口中共同操作，因此用户需要首先将历史窗口嵌回到MATLAB的操作界面中。

◆ **运行命令行**：这个操作的功能是运行原来输入的命令行，得到原来命令行的结果。在历史窗口中选择需要运行的历史命令行，然后单击鼠标右键，在弹出的快捷菜单中选择“Evaluate Selection”选项，如图1.20所示。

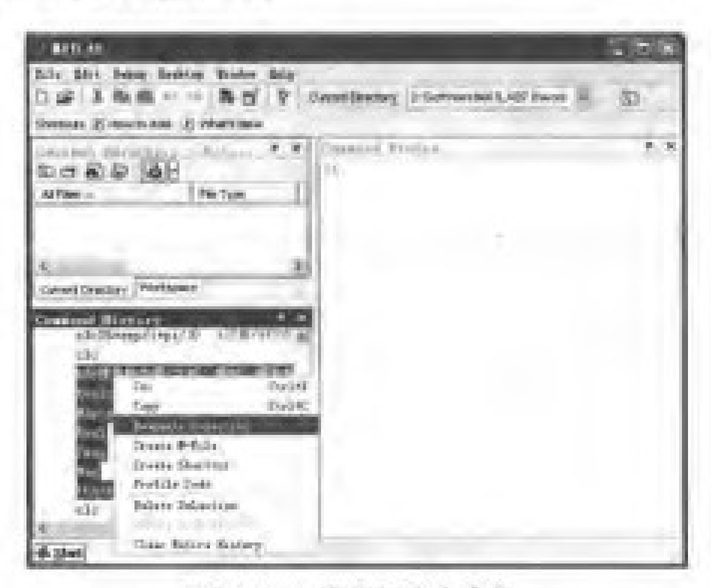


图1.20 运行历史命令行

运行历史命令行后，在命令窗口中就会显示相应的运行结果，如图1.21所示。



图1.21 运行的结果



如果用户运行的是单行命令行，则可以直接使用鼠标左键来双击命令行；如果用户运行的是多行命令行，用户则需要按下“Ctrl”键同时选择多行命令行。

✦ **创建M文件**：用户可以根据需要将历史命令行编写成为M文件。在历史窗口中选择需要运行的历史命令行，然后单击鼠标右键，在弹出的快捷菜单中选择“Create M-File”选项，如图1.22所示。

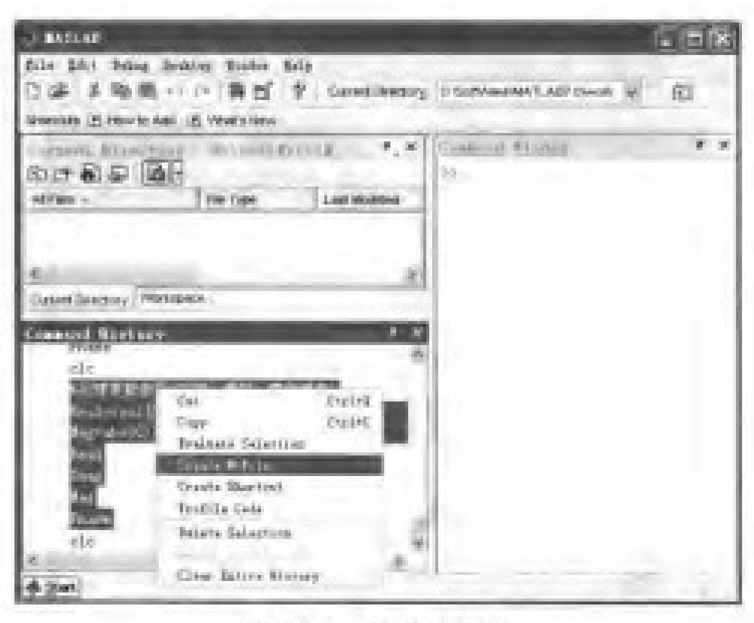


图 1.22　创建 M 文件

选择相应的菜单选项后，MATLAB就会调用M文件编辑器，并且将用户选择的历史命令行填写在M文件编辑器中，如图1.23所示。

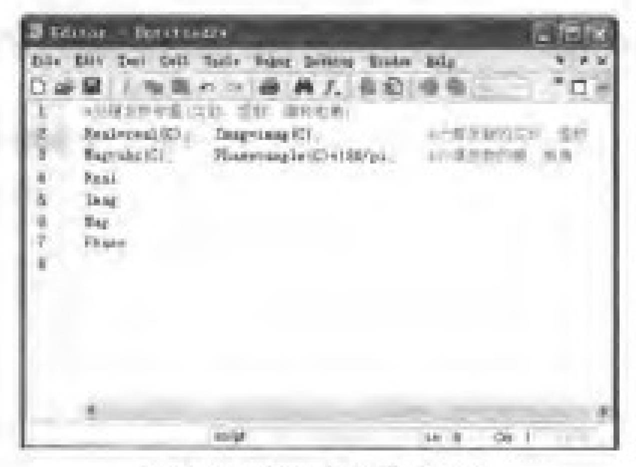


图 1.23　创建完成的 M 文件



M文件是MATLAB的重要应用之一，在后面的章节中将详细介绍如何使用和编辑M文件。当用户通过历史命令行创建M文件后，需要经过必要的操作才能得到完整的M文件。

1.3.10 使用实录命令

在 MATLAB 中，为用户提供 diary 命令，用来创建“日志”文件。在这个“日志”文件中，记录了当前命令窗口中的所有内容，包括命令和计算结果等。文件的保存格式是 ACSII，因此用户需要使用“记事本”或者其他文本软件来阅读日志文件。

在 MATLAB 程序启动的前提下，首先单击操作界面中的“Current Directory”按钮旁边的浏览按钮，在打开的对话框中选择合理的路径。在后面步骤中创建的日志文件会保存在该路径中。如果用户不修改路径，MATLAB 会将日志文件保存在默认的 MATLAB7.0\work 文件夹中。

在用户修改保存路径后，就可以在命令窗口中输入关于实录的命令，创建日志文件。下面举例详细介绍。

例 1.8 在 MATLAB 中创建名称为“first_diary”的日志文件，并阅读该日志文件，如图 1.24 所示。



图 1.24 创建日志文件

上面的详细程序清单如下：

```
>> diary first_diary    %创建名称为 first_diary 的日志文件
>> %显示如何在 MATLAB 中输入复数
z1=8+10i;               %直接按照直角坐标的方式输入
z2=12+6*i;              %运算符构成的直角坐标的方式输入
z3=25*exp(i*pi/3);      %运算符构成的极坐标的方式输入
A=[z1,z2,z3]

A =

     8 +        10i           12 +        6i           12.5 +    21.6511
>> diary off            %关闭记录命令，完成日志文件
```

由于用户没有修改保存路径，因此可以在 MATLAB7.0\work 路径中找到相应的日志文件，并用“记事本”软件打开该日志文件，如图 1.25 所示。



使用上面的方法创建的日志文件不带有扩展名，一般可以使用文本软件或者 MATLAB 的 M 文件编辑器来阅读和编辑该日志文件。

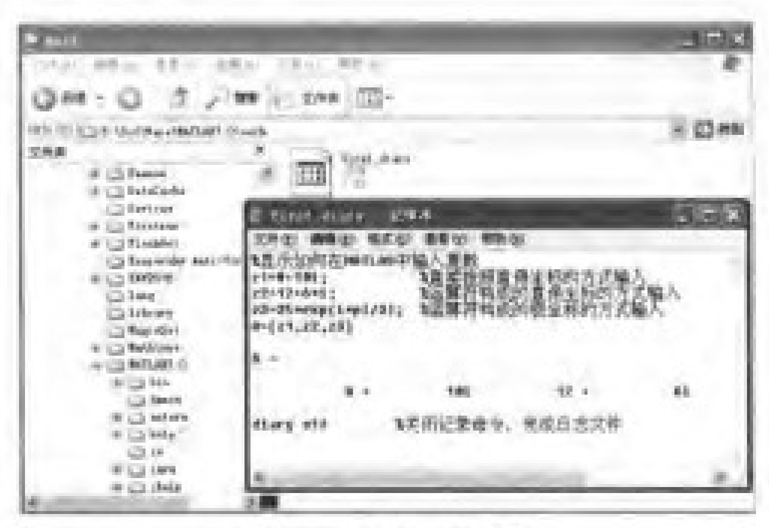


图 1.25　阅读日志文件

1.3.11　当前目录管理器和路径管理

在MATLAB中包含大量的函数和数据库。当用户在命令窗口中输入命令行的时候，MATLAB如何搜索到相应的函数和数据，怎样有效地管理这些函数和数据库的路径，提高搜索的效率是本小节的核心问题。

同时，用户在使用MALTAB的时候，也会产生大量的MATLAB文件，如何管理好这些文件也是一个重要的问题。如果不能有效管理这些文件，将会直接影响用户运行MATLAB的效率。

在MATLAB中，提供当前目录浏览器来管理各种文件。在默认的情况下，当前目录浏览器在MATLAB操作界面的左上方的后台，单击"Current Directory"选项卡，可以使目录浏览器在MATLAB的前台显示。然后单击目录浏览器右上方的按钮，可以查看目录管理器的详细外观，如图1.26所示。

图 1.26　当前目录浏览器

在默认情况下，当前目录浏览器包括菜单栏、当前目录设置区、工具菜单和文件详细列表等。其中，用户需要经常使用的是文件详细列表区域。在该区域中，用户可以运行或者编辑M文件，装载MAT数据文件等常见操作，如图1.27所示。

上面的操作过程十分简单，用户首先在当前目录浏览器中选择相应的M文件，然后单击数据右键，在弹出的快捷菜单中选择对应的选项。例如，用户希望运行对应的M文件，可以选择"Run"选项；用户希望编辑该M文件，则选择菜单中的"Open"选项，该M文件就会出现在M文件的编辑器中。其他的操作都可以在该菜单中选择对应的选项。

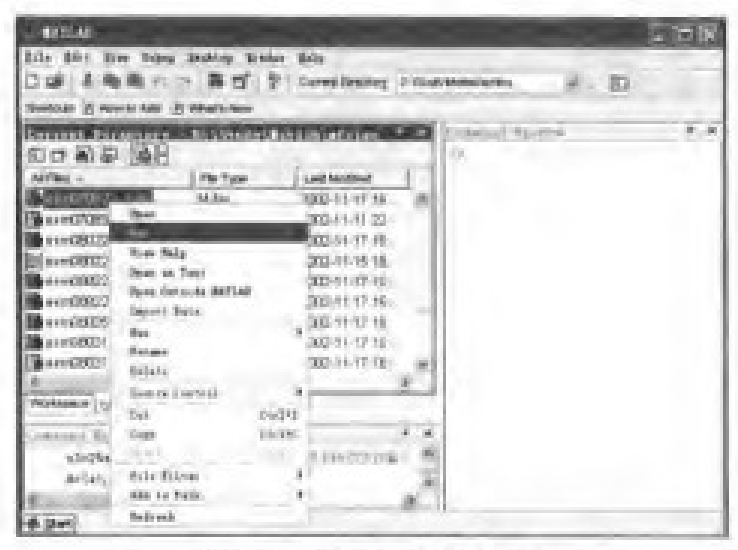


图 1.27 操作和编辑 M 文件

1.3.12 设置当前目录

在默认的情况下，启动 MATLAB 的时候，系统会将当前目录设置为 "MATLAB7.0\work" 或者 "MATLAB7.0\bin\win32"，这取决于用户启动 MATLAB 的方式，这在前面的内容中已经介绍。

对于 "MATLAB7.0\work" 这个目录路径，用户在该路径中存放用户的文件是允许的而且是安全的，因此用户可以沿用这个默认路径；而对于 "MATLAB7.0\bin\win32" 这个目录路径，则不建议用户使用，如果用户启动 MATLAB 的时候产生了这个路径，建议用户改变这个默认的目录路径。

尽管用户可以沿用 "MATLAB7.0\work" 这个目录路径，但是根据笔者经验，为了方便用户管理各种 MATLAB 文件，还是建议用户创建自己的工作路径，存放自己创建的应用文件。而将 "MATLAB7.0\work" 这个目录路径作为临时目录使用。

创建工作目录的方法和在 Windows 中创建目录的方法完全相同，用户可以参阅相应的书籍。建议将用户创建的工作目录设置为当前目录，这是因为在 MATLAB 环境中，如果不特别指明存放目录，MATLAB 都会默认地将文件存放在当前目录中。如果用户将自己设置的工作目录设置为当前目录，就可以保证 MATLAB 运行的可靠和便捷。

用户可以在当前目录浏览器中的 "目录设置框" 中输入新的工作目录，或者单击该界面中的 "目录浏览键"，选择新的工作目录，如图 1.28 所示。

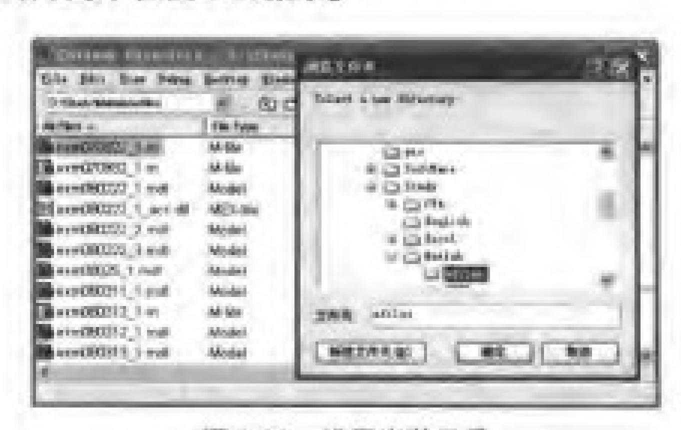


图 1.28 设置当前目录

除了上面的方法外，习惯了编程的用户也可以在命令窗口中输入控制命令来修改当前目录。这

种方法适用于MATLAB的各个版本中，而且控制命令不仅可以在命令窗口中使用，也可以在M文件中使用，效果相同。设置当前目录的控制命令是cd，例如，用户需要将当前目录设置为D:\Study\Matlab\mfiles，对应的控制命令为 cd D:\Study\Matlab\mfiles。



使用上面两种方法设置的当前目录，只能在当前开启的MATLAB环境有效，如果用户重新设置MATLAB应用程序，上面的设置操作需要重新进行。

1.3.13 MATLAB 的搜索路径

在MATLAB中，所有的文件都被存放在一组结构比较严谨的目录路径中。MATLAB会将这些目录按照一定的次序设置为搜索路径的各个节点。当用户在运行MATLAB的时候，程序就会沿着这个设定的路径进行搜索，查找相应的文件、函数或者具体的数据。

当MATLAB进行搜索的时候，会按照一定的次序进行，例如用户在命令窗口中输入命令my_matlab，程序首先在内存中搜索是否有名为my_matlab的变量，如果没有找到，则再检查my_matlab是否是内置的函数，如果没有搜索到结果，则在当前目录中检查是否有名为my_matlab的M文件;如果还是没有搜索结果，则在其他目录中检查是否存在名为my_matlab的M文件。

上面的搜索过程就是MATLAB的典型搜索路径。了解这个搜索路径，用户就可以有效地进行路径管理，提高搜索效率和MATLAB的运行效率。

如果用户有多个目录需要同时和MATLAB进行信息交换，用户应该将这些目录设置在搜索路径中，这些目录中的所有内容就都可以被MATLAB调用。或者，用户设置了某个目录来存放所有的文件和数据，这个目录也应该被设置在MATLAB的搜索路径中。

用户可以在MATLAB的命令窗口中输入"pathtool"命令或者选择"File"⇨"Set Path"命令，打开"Set Path"对话框，在其中设置路径的各个参数，如图1.29所示。

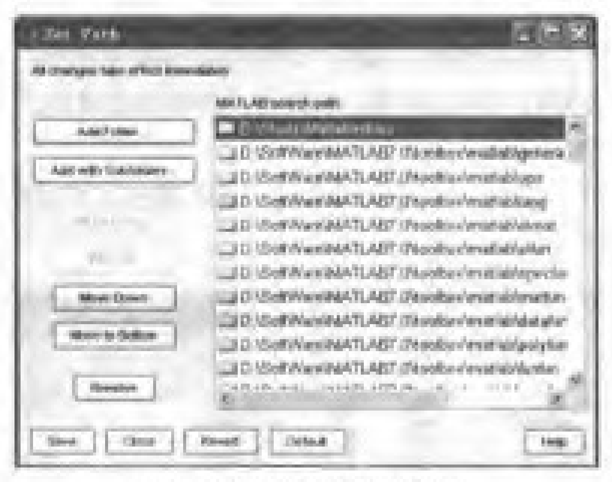


图 1.29　路径设置对话框

在上面的对话框中，单击"Add Folder"按钮，然后选择相应的路径，就可以将该路径"D:\Study\Matlab\mfiles"设置到搜索路径中。在上面的路径设置对话框中，用户可以编辑任意一个现存的路径信息，例如，添加子文件夹、移动文件夹的位置或者删除路径等，只需单击对话框中的对应按钮。



在上面的路径设置对话框中，存在两种修改状态。如果用户在修改路径信息的时候，仅使用对话框中的左侧按钮，那么这种修改行为只是当前有效；如果用户在修改路径信息的时候，单击了对话框中的"Save"按钮，则这种修改行为将永久有效。

除了上面的设置方法，MATLAB还提供path命令来设置路径，这个命令对于MATLAB的各个版本都适用。例如，可以使用path命令查看MATLAB的路径信息，如图1.30所示。

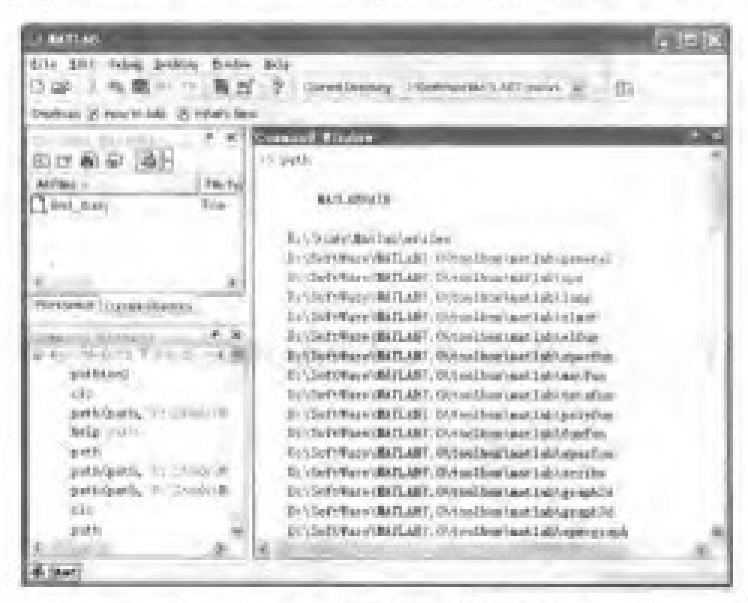


图 1.30 查看MATLAB的路径信息



使用path命令完成的路径修改信息只在当前MATLAB的环境中有效。当用户重新启动MATLAB后，使用该命令完成的路径信息都将失效。

1.3.14 工作空间浏览器和数组编辑器

在默认的情况下，工作空间浏览器位于MATLAB操作界面的左上侧的后台，单击"Workspace"选项卡，可以使目录浏览器在MATLAB的前台显示。然后单击目录浏览器右上方的按钮，可以查看工作空间浏览器的详细外观，如图1.31所示。

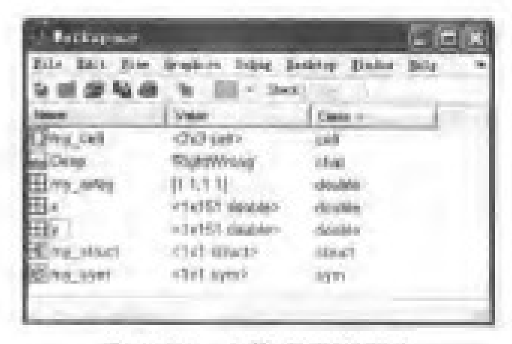


图 1.31 工作空间浏览器

和其他MATLAB组件的界面相比，工作空间浏览器的菜单栏中多了一个"Graphics"选项，当用户选中其中某个变量时，用户可以选择该菜单选项中的子选项，很方便地绘制相应的各种图形，如图

1.32 所示。



图 1.32　图形选项菜单

在上面的图表选项菜单中，用户可以选择各种常见的图表类型。如果菜单选项中没有合适的图表类型，用户可以选择菜单中的 "More Plots" 选项，打开 "Plot Catalog" 对话框，选择合适的图表类型，如图 1.33 所示。



图 1.33　图形分类对话框

上面的 "Plot Catalog" 对话框中的左侧是图形的分类，中间表示的是图表类型，右侧则是用户选择的图表类型的描述。



工作空间浏览器中的 "Graphics" 菜单选项是 MATLAB7.0 中新增的，在 MATLAB6.x 版本中，对应界面中并没有该菜单选项。

除了非常强大的图形绘制功能之外，工作空间浏览器还有其他多种应用功能，例如内存变量的查阅、保存和编辑等。所有这些操作工作都比较简单，只需要在工作空间浏览器中选择相应的变量，然后右击鼠标，在弹出的快捷菜单中选择相应的菜单选项，如图 1.34 所示。

对变量的操作工作可以从菜单选项中的名称看出，例如，"Rename" 菜单选项表示对应的操作是重命名该变量；"Copy" 菜单选项表示对应的操作是复制该变量等。



图 1.34 编辑内存变量

1.3.15 变量的编辑命令

在 MATLAB 中，用户除了可以在工作空间浏览器中编辑内存变量之外，还可以在 MATLAB 的命令窗口中输入相应的命令，查阅和删除内存变量。下面用简单的案例，说明如何在命令窗口中对变量进行操作。

例 1.9 在 MATLAB 命令窗口中查阅内存变量。

具体步骤如下：

在命令窗口中输入 who 和 whos 命令，查看内存变量的信息，如图 1.35 所示。

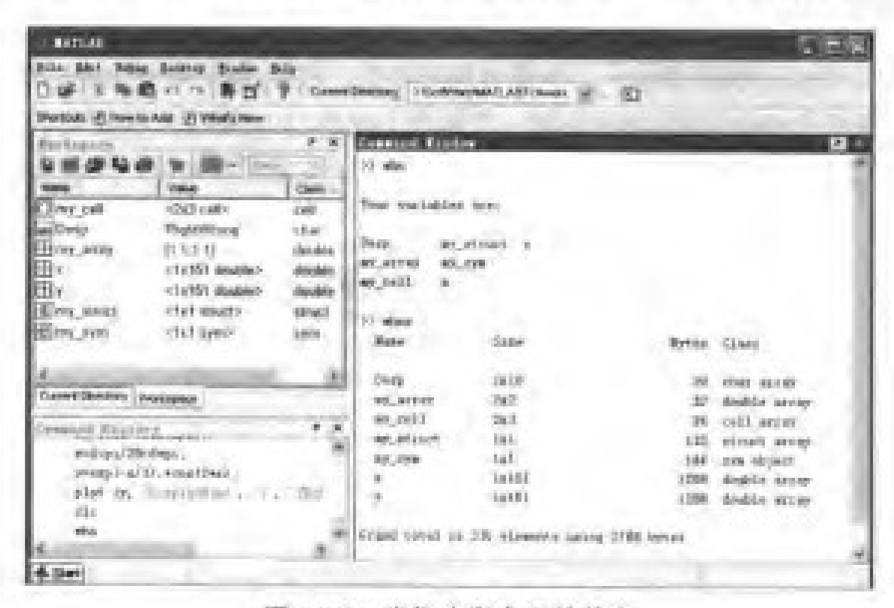


图 1.35 查阅内存变量的信息



who 和 whos 命令适用于 MATLAB 的各个版本，两个命令的区别只在于内存变量信息的详细程度。两个命令结果的列表次序随具体情况而不同。

例 1.10　承接上面的例子，在 MATLAB 命令窗口中删除内存变量 Desp。

在命令窗口中输入下面的命令行：

```
>> clear Desp;
>> who
```

得到的结果如图 1.36 所示。

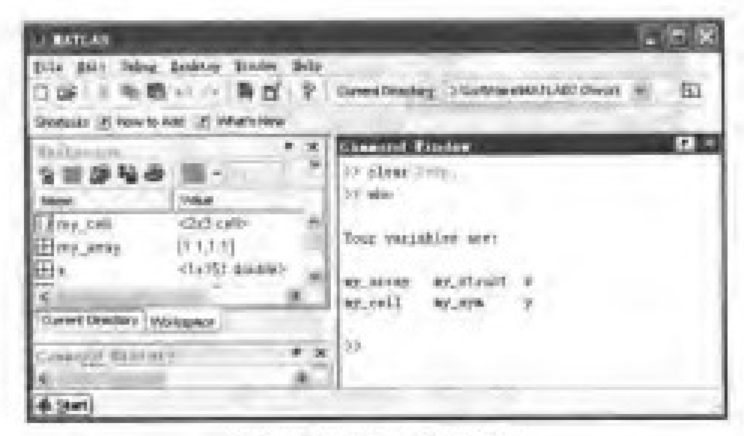


图 1.36　删除内存变量

和前面的例子对比，用户可以看出，当用户运行clear命令后，将Desp变量从工作空间删除，而且在工作空间浏览器中也将该变量删除。



如果用户在命令窗口中直接输入clear命令，就可以删除工作空间中的所有变量；如果用户需要删除多个内存变量，可以在clear命令后面依次添加删除的变量名称。

1.3.16　数组编辑器

在 MATLAB 中，数组和矩阵都是十分重要的基础变量，因此 MATLAB 专门提供数组编辑器这个工具来编辑数组。选择工作空间浏览器中任意一个数组（就是 class 类别为 double 的内存变量），然后选择菜单栏中的"Open Selection"选项，或者直接双击该变量，就可以打开该变量的数组编辑器，如图 1.37 所示。

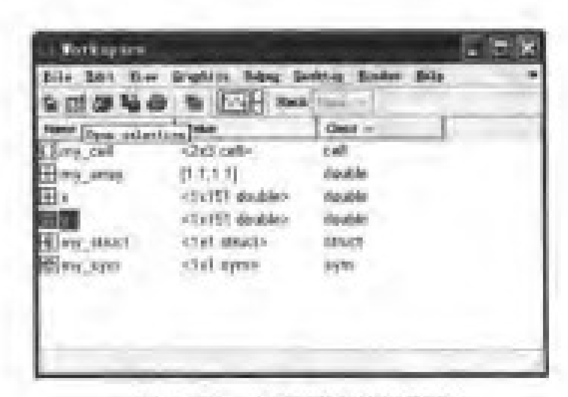


图 1.37　打开数组编辑器



在MATLAB中，数组编辑器只支持一维或者二维数值数组，而不支持元胞数组、构架数组、符号类数组、三维数组、字符串数组等。因此，在前面的步骤中，必须选择数值数组打开相应的数组编辑器。

在上面的步骤中，用户打开了变量A的数组编辑器，如图1.38所示。



图 1.38 变量的数组编辑器

用户可以在数组编辑器中直接编辑该变量。对于大型数组，使用数组编辑器会给用户带来很大的便利。

1.3.17 存取数据文件

在MATLAB中，提供save和load命令来实现数据文件的存取。表1.6列出了命令的常见用法。

表 1.6 MATLAB 文件存取的命令

命令	功能
Save Filename	将工作空间中的所有变量保存到名为Filename的MAT文件中
Save Filename x y z	将工作空间的x、y、z变量保存到名为Filename的MAT文件中
Save Filename -regexp pat1 pat2	将工作空间中符合表达式要求的变量保存到Filename.mat文件中
Save Filename x y z- ASCII	将工作空间的x、y、z变量保存到名为Filename的8位ASCII文件中
Load Filename	将名为Filename的MAT文件中的所有变量读入内存
Load Filename x y z	将名为Filename的MAT文件中的x、y、z变量读入内存
Load Filename -regexp pat1 pat2	将名为Filename的MAT文件中符合表达式要求的变量读入内存
Load Filename x y z- ASCII	将名为Filename的ASCII文件中的x、y、z变量读入内存

上表中列出了几个常见的文件存取命令，用户可以根据需要选择对应的存取命令；对于一些比较少见的存取命令，用户可以查阅MATLAB的相关帮助。



在上面的命令中，参数Filename可以带有路径，但是不能带扩展名；x、y、z代表变量名称，个数不限，但是名称之间必须以空格来分隔；- ASCII参数表示数据将以ASCII格式来处理，生成的文件可以使用"文字处理器"来编辑，一般适用于数据较多的文件；如果命令行后面没有 - ASCII参数，在默认情况下，数据将以二进制格式来处理，生成以mat扩展名的文件。

在MATLAB中，除了可以在命令窗口中输入相应的命令之外，也可以在工作空间中选择相应的按钮，实现数据文件的存取工作。例如，用户可以选择工作空间浏览器中的"Files" ⇨ "Save Workspace

As" 命令，将所有变量保存到 mat 文件中，如图 1.39 所示。

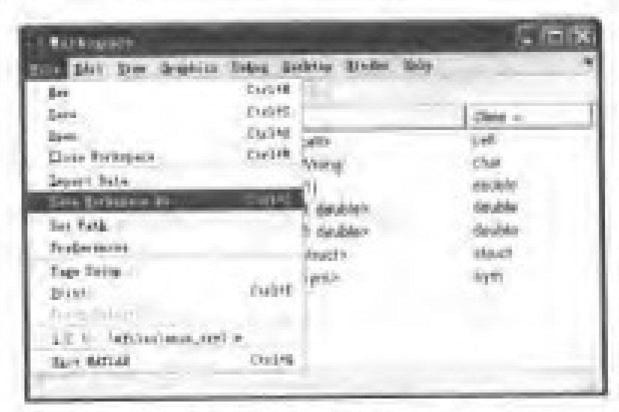


图 1.39　保存所有的变量

当用户选择 "Save Workspace As" 命令后，可以打开 "Save to MAT-File" 对话框，用户可以在对话框中输入数据文件的名称，然后选择保存路径，就可以保存所有的变量。

如果用户需要保存部分变量，可以在工作空间浏览器中同时选择需要保存的变量，然后单击鼠标右键，在弹出的快捷菜单中选择 "Save As" 命令，将用户选择的变量保存到 mat 文件中，如图 1.40 所示。

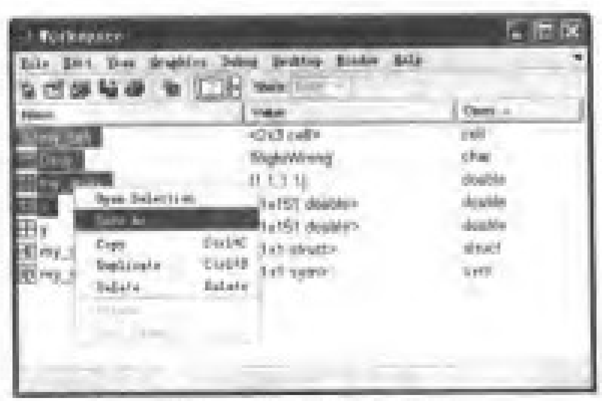


图 1.40　保存部分变量



当用户选择 "Save As" 命令后，同样可以打开 "Save to MAT-File" 对话框，用户可以在对话框中输入数据文件的名称，然后选择保存路径，就可以保存所选择的变量。

在 MATLAB 的工作空间浏览器中，用户也可以加载数据。选择工作空间浏览器中的 "Files" ⇨ "Open" 命令，打开 "Open" 对话框，加载数据文件中的所有变量。同时可以选择 "Files" ⇨ "Import Data" 命令，打开 "Open" 对话框，选择需要加载的数据文件，然后单击 "打开" 按钮，打开 "Import Wizard" 对话框。用户可以在打开的 "Import Wizard" 对话框中勾选希望加载的变量，如图 1.41 所示。

在 "Import Wizard" 对话框中，左侧表明的就是数据文件中的变量，用户可以在这些变量中选择需要加载的变量，对话框的右侧则是用户所选变量的预览效果。在图中，用户选择的是变量 "my_array"，在右侧显示该变量的预览效果。



如果某些数据是经过很复杂的计算过程得到的，为了避免再次的重复计算，通常会使用 Save 命令来保存。如果在后面的操作中需要使用这些数据，则可以使用 Load 命令来加载。

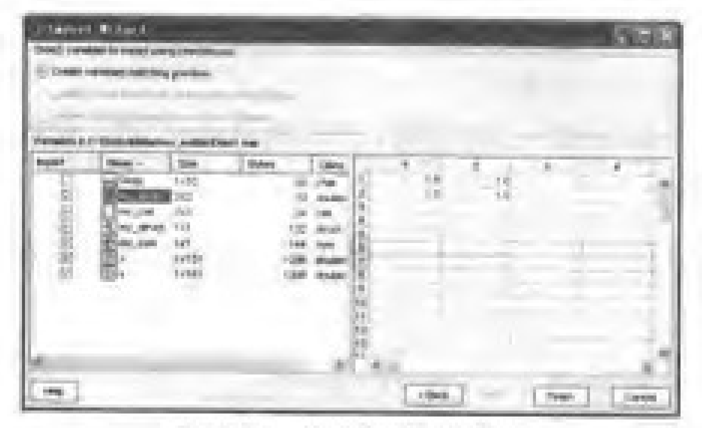


图 1.41 加载变量的对话框

1.4 MATLAB7.0 的帮助系统

MATLAB的各个版本都为用户提供非常详细的帮助系统，可以帮助用户更好地了解和运用MATLAB。因此，不论用户是否使用过MATLAB，是否熟悉MATLAB，都应该了解和掌握MATLAB的帮助系统。同时，在MATLAB6.x版本以后，帮助系统的帮助方式、内容层次相对于之前的版本发生了本质变化，因此，用户更加有必要了解MATLAB7.0的帮助系统。在本节中，将详细介绍MATLAB7.0的帮助系统。

1.4.1 纯文本帮助

在MATLAB中，所有执行命令或者函数的M源文件都有较为详细的注释。这些注释都是用纯文本的形式来表示的，一般都包括函数的调用格式或者输入参数、输出结果的含义。这些帮助是最原始的（相当于最低层的源文件），当MATLAB不同版本中函数发生变化的时候，这些文本帮助也会同步更新。

下面使用简单的例子来说明如何使用MATLAB的纯文本帮助。

例 1.11 如何在MATLAB中查阅帮助信息。

根据MATLAB的帮助体系，用户可以查阅不同范围的帮助，具体步骤如下。

step 1 在MATLAB的命令窗口中输入help help命令，然后按"Enter"键，查阅如何在MATLAB中使用help命令，如图1.42所示。

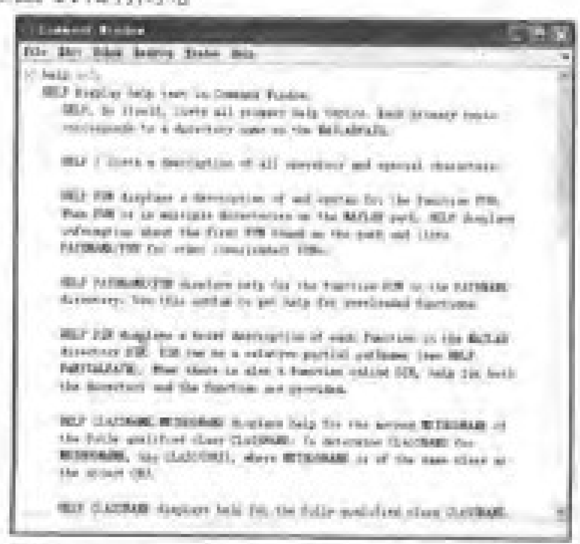


图 1.42 使用help命令的帮助信息

上面的操作界面显示了如何在MATLAB中使用help命令的帮助信息，用户可以详细阅读上面的信息来解如何使用help命令。

step 2 在MATLAB的命令窗口中输入help命令，然后按"Enter"键，查阅关于MATLAB系统中的所有主题的帮助信息，如图1.43所示。



图 1.43　查阅关于主题的帮助信息

step 3 在MATLAB的命令窗口中输入help topic命令，然后按"Enter"键，查阅关于该主题的所有帮助信息，如图1.44所示。

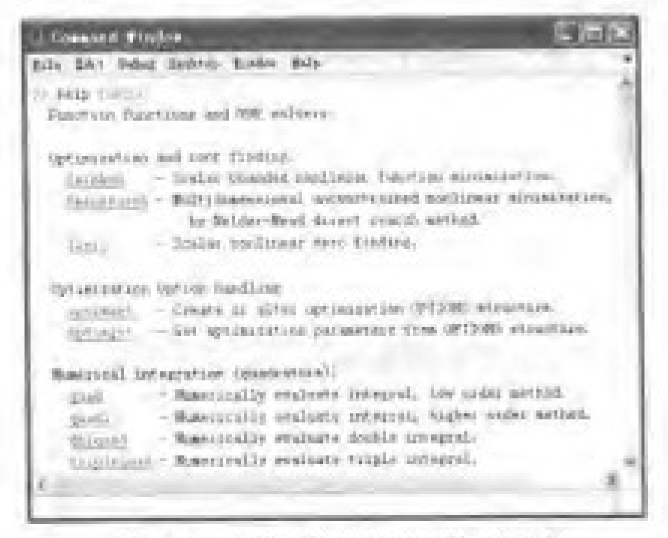


图 1.44　查阅主题下的函数帮助信息



上面的步骤中，简单地演示了如何在MATLAB中使用help命令，获得各种函数、命令的帮助信息。在实际应用中，用户可以灵活使用这些命令来搜索所需的帮助信息。

例 1.12　如何在MATLAB中搜索各命令的帮助信息，在M函数文件中搜索包含关键字jacobian的所有M函数文件名，如图1.45所示。

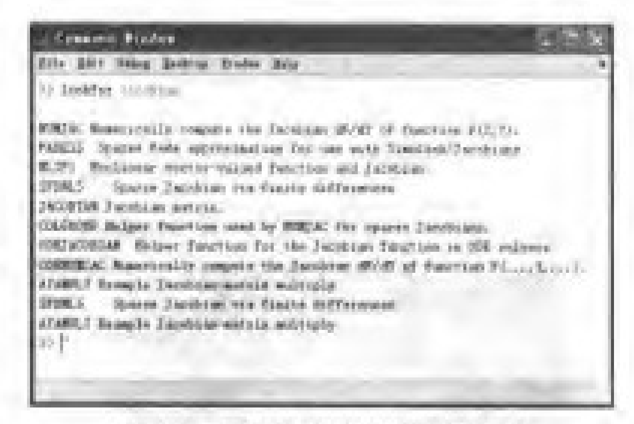


图 1.45　搜索 M 文件的帮助文件

1.4.2 演示（demo）帮助

在MATLAB中，各个工具包都有设计好的演示程序，这组演示程序在交互界面中运行，操作非常简便。因此，如果用户运行这组演示程序，然后研究演示程序的相关M文件，对MATLAB用户而言是十分有益的。这种演示功能对提高用户对MATLAB的运用能力有着重要的作用。特别对于那些初学者而言，不需要了解复杂的程序就可以直观地查看程序结果，可以加强用户对MATLAB的掌握能力。

在MATLAB的命令窗口中输入"demo"命令，就可以调用关于演示程序的帮助对话框，如图1.46所示。



图 1.46　MALTAB 中的 demo 帮助

在上面的对话框中，用户可以在对话框的左侧选择演示的内容，例如选择"Graphics"选项，在对话框的右侧会出现该项目下的各种类别的演示程序。在本例中，单击上面对话框中的"3-D Surface Plots"选项，MATLAB对话框中会显示关于"3-D Surface Plots"演示程序的介绍，然后单击对话框中的"Run this demo"选项，MATLAB会打开"R3-D Plots in Handle Graphics"对话框，该对话框就是演示demo的交互界面，如图1.47所示。



图 1.47　显示 demo 的交互界面

演示该demo的是一个交互界面，用户可以调整该界面中选项按钮的数值，改变图形的显示参数，这些修改的参数会出现在 "Command Window" 选框中，如图 1.48 所示。

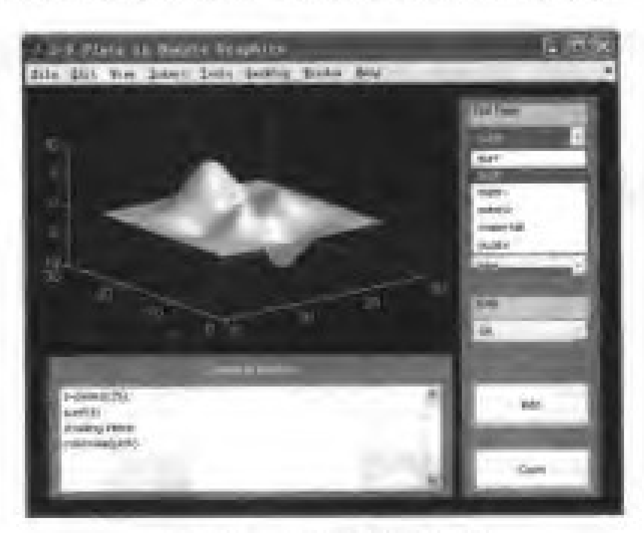


图 1.48　动态演示 demo

用户除了可以在打开的动态界面中演示 demo 之外，还可以查看该 demo 的程序代码，单击 "Help" 对话框中的 "View code for graf3d" 选项，查看该 GUI 界面的 MATLAB 程序代码，如图 1.49 所示。



除了在命令窗口中输入 demo 命令来调用 demo 的 "Help" 对话框之外，还可以选择 MATLAB 的操作界面中的 "Start" ⇨ "Demos" 命令，调用 demo 的对话框。



图 1.49 查看 demo 的程序代码

1.4.3 帮助导航 / 浏览器

在MATLAB中提供帮助信息的"导航/浏览器"交互界面是MATLAB6.x以后的版本的重要改进，这个交互界面主要由帮助导航器和帮助浏览器两个部分组成。这个帮助文件和M文件中的纯文本帮助无关，而是MATLAB专门设置的独立帮助系统。该系统对MATLAB的功能叙述得全面、系统，而且界面友好，使用方便，是用户查找帮助的重要途径。

用户可以在MATLAB的命令窗口中输入命令"helpbrowser"或者"helpdesk"，或者在操作桌面中单击 ? 按钮，打开"帮助导航 / 浏览器"交互界面，如图 1.50 所示。

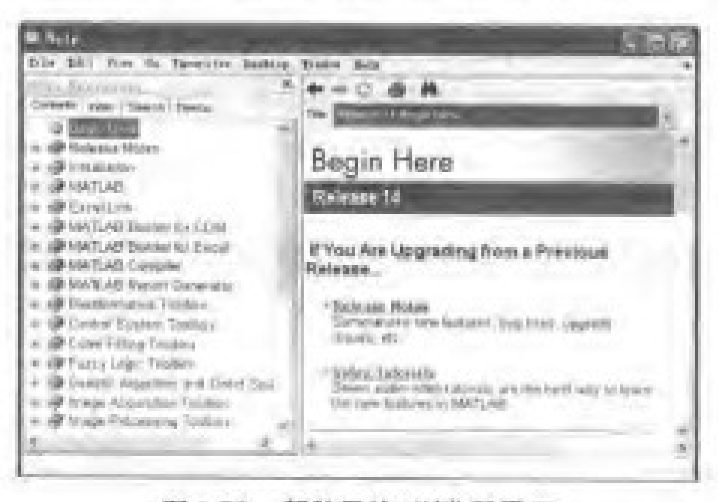


图 1.50 帮助导航 / 浏览器界面

1.4.4 Contents 帮助文件目录窗

在默认情况下，当用户在MATLAB中打开"帮助导航/浏览器"交互界面时，界面会选择"Contents"选项卡。这个窗口中使用节点可展开的目录树来列出各种帮助信息，直接使用鼠标单击相应的目录条，就可以在浏览器中显示相应标题的HTML 帮助文件。

这个窗口是向用户提供全方位系统帮助的向导，层次清晰、功能划分规范，用户可以查找相应的帮助信息。例如，初学用户希望了解MATLAB，可以选择对话框中的"MATLAB" ⇨"Getting Started" ⇨ "Introduction" ⇨ "What is MATLAB?" 选项，在浏览器中查看关于MATLAB的HTML帮助文件，如图1.51所示。

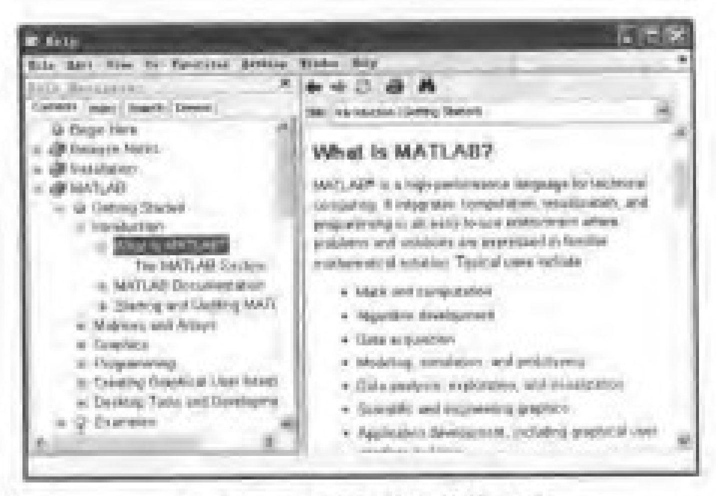


图1.51 查看帮助文件的目录



在MATLAB的帮助浏览器中，除了提供HTML类型的帮助文件外，还提供PDF类型、Web类型的帮助文件。对于不同类型的文件，在目录树中对应文件名称之前的符号会有不同。

1.4.5 Index 帮助文件索引窗

在MATLAB中，为了提高用户使用帮助文件的效率，专门为命令、函数和一些专用术语提供索引表。用户可以选择交互界面中的"Index"选项卡，然后在"Search index for"选框中输入需要查找的名称，在其下面就会出现与此匹配的词汇列表。同时，在浏览器的界面显示相应的介绍内容。例如，在"Search index for"选框中输入"sin"进行搜索，得到的结果如图1.52所示。

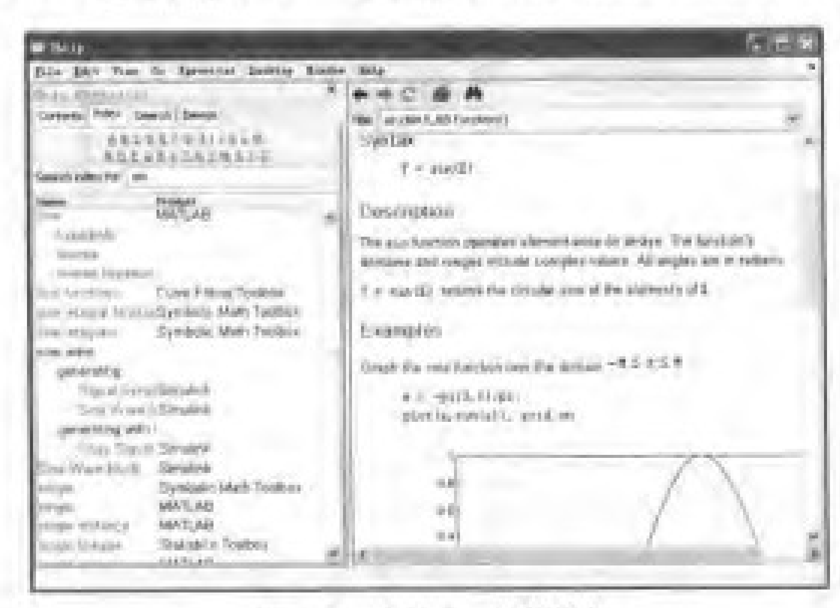


图1.52 查看"sin"的信息



在MATLAB7.0中，交互界面的"Index"选项中添加了A、B、C.....Z的字母索引体系，当用户单击其中一个字母的时候，会在其下面的列表中显示含有该字母的资源信息。

1.4.6 Search帮助文件搜索窗

和前面"Index"选项卡不同，在"Search"选项卡中，用户可以利用关键词在全文中查找到与关键字相匹配的内容。在"Search"选项卡中的搜索范围是整个HTML文件而不仅仅是专业术语，因此其覆盖面更加广泛。

在默认的情况下，该选项卡中的搜索方式是全文搜索。例如，在"Search for"选框中输入"laplace"，就可以在整个HTML文件中搜索含有该字母片断的内容，如图1.53所示。



图1.53 在"Search"窗口中查看"laplace"的信息

在搜索结果的底部，有"Search Support Database on Web for laplace"的超链接文字，单击该文字，MATLAB就会启动IE在相关站上搜索关于laplace的资源信息。同时，在搜索界面的最底部，显示"18 pages contain the word:laplace"的字样，表明搜索的结果个数。

在帮助浏览器中会显示每个搜索结果的HTML文件，同时将关键字"laplace"高亮显示，方便用户查阅相应的信息。



在MATLAB6.5以后的版本中，MATLAB开始支持关键字之间的逻辑运算符AND、OR、NOT等，这些逻辑运算符都需要大写，同时必须和关键字之间保持间隔。

1.5 小结

在本章中，首先向用户介绍了MATLAB软件的特点，然后循序渐进地介绍了如何安装MATLAB7.0以及MATLAB7.0的工作环境、帮助系统等内容。希望通过本章的内容，用户能够对MATLAB有一个直观的印象。在后面的章节中，将详细介绍关于MATLAB的基础知识和基础操作方法。

第2章　MATLAB基础知识

本章包括

- 创建数值数组
- 操作数值数组
- 稀疏矩阵
- 构架数组
- 字符串数组

数组或者矩阵都是MATLAB的基础内容，几乎所有的数据都是用数组的形式进行储存的，因此，MATLAB又被称为矩阵实验室。从MATLAB5.x版本开始，基于面向对象的考虑，这些数组就成为MATLAB中的内建数据类型（Built-in Data Type），而数组运算就是定义这些数据结构上的方法。在本章中，将介绍关于数组类型和数组运算的内容。

创建数值数组

创建数组是所有MATLAB运算和操作的基础。针对不同维度的数组，MATLAB提供多种创建方法，可以分别创建不同要求的数组类型。在本节中，将分别根据数组维度以及方法的不同，介绍如何创建数组。

一维数组的创建方法

在MATLAB中，一维和二维数组都被认为是比较低维的数组。它们的创建方法比较简单，同时，也是创建高维数组的基础条件，在本小节中，将以简单的例子来说明如何在MATLAB中创建各种不同的数组类型。

例2.1　在MATLAB中，使用不同的方法来创建一维数组。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>> data1=[pi;log(5);7+6;2^3];
>> data2=[pi log(5) 7+6 2^3];
>> data3=2:2:10;
>> data4=2:10;
>> data5=linspace(2,10,5);
>> data6=logspace(1,5,10);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
data1 =
    3.1416
    1.6094
   13.0000
    8.0000
data2 =
    3.1416    1.6094   13.0000    8.0000
```

```
data3 =
     2     4     6     8    10
data4 =
     2     3     4     5     6     7     8     9    10
data5 =
     2     4     6     8    10
data6 =
  1.0e+005 *
    0.0001    0.0003    0.0008    0.0022    0.0060    0.0167    0.0464
0.1292    0.3594    1.0000
```

上面的结果基本演示了在MATLAB中创建一维数组的方法。

- **直接输入法**：在上面的程序代码中，data1和data2就是直接输入法。其中data1在输入数据的时候，使用了分号，创建了一维列数组；data2则在创建过程中使用了空格，因此创建了一维行数组。
- **步长生成法**：在上面的程序代码中，data3和data4的创建方法就是步长生成法，其通用方法是a:inc:b，其中a表示的是数组的第一个元素，inc是创建数组之间的间隔，也就是步长，b则是数组中的最后一个元素。其中inc可以省略，默认的数值为1。
- **定数线性采样法**：在上面的程序代码中，data5的创建方法就是定数线性采样方法。该方法在设定的"总个数"的条件下，均匀采样分布生成一维行数组。这种方法的调用格式为：x=linspace(a,b,n)，其中a、b分别是数组的第一个和最后一个元素，n表示的是采样点数。
- **定数对数采样法**：在上面的程序代码中，data6的创建方法就是定数对数采样法。这种方法在设定的"总个数"的条件下，经过"常用对数"采样生成一维行数组。这种方法的调用格式为x=logspace(a,b,n)。



在上面的步长生成法中，步长参数inc的数值可以是正数，也可以是负数，当inc是正数的时候，必须满足a<b；当inc是负数的时候，必须满足a>b。

2.1.2 二维数组的创建方法

在本小节中，将介绍如何在MALTAB中创建二维数组。

例2.2 在MATLAB中创建二维数组。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>> Data1= [1     2     3
           4     5     6
           7     8     9];
>> Data2=[1,2,3;4,5,6;7,8,9];
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
Data1 =
     1     2     3
     4     5     6
     7     8     9
Data2 =
     1     2     3
```

```
    4     5     6
    7     8     9
```

上面的例子基本演示在MATLAB中创建二维数组的方法，Data1 的创建方法比较直接，得到的结果就是输入的结果；Data2 的创建方法则是更加普遍的方法，关于该方法需要注意下面的内容：

- 整个输入数组必须以方括号“[]”作为创建的首尾；
- 数组的行和行之间必须用分号“;”来间隔；
- 数组的列和列之间必须用逗号“,”来间隔。



上面使用的创建方法适用的范围比较狭窄，当数组的行或者列数比较大的时候，上面的创建方法就会比较烦琐。至于其他方法将在后面章节中介绍。

2.1.3 使用下标创建三维数组

在MATLAB中，习惯将二维数组的第一维称为“行”，第二维称为“列”，而对于三维数组，其第三维则习惯性地称为“页”。在MATLAB中，将三维或者三维以上的数组统称为高维数组。由于高维数组的形象思维比较困难，在本小节中将主要以三维为例来介绍如何创建高维数组。

例 2.3 使用下标引用的方法创建三维数组。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
A(2,2,2)=1;
>> for i=1:2
for j=1:2
for k=1:2
A(i,j,k)=i+j+k;
end
end
end
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>> A(:,:,1)
      3     4
      4     5
>> A(:,:,2)
      4     5
      5     6
```

step 3 创建新的高维数组。在MATLAB的命令窗口中输入下面的程序代码：

```
>> B(3,4,:)=2:5;
```

step 4 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>> B(:,:,1)
      0     0     0     0
      0     0     0     0
      0     0     0     2
>> B(:,:,2)
```

```
     0      0      0      0
     0      0      0      0
     0      0      0      3
>> B(:,:,3)
     0      0      0      0
     0      0      0      0
     0      0      0      4
>> B(:,:,4)
     0      0      0      0
     0      0      0      0
     0      0      0      5
```



从上面的结果中可以看出，当使用下标的方法来创建高维数组的时候，需要使用各自对应的维度的数值，没有指定的数值则在默认情况下为0。

2.1.4 使用低维数组创建三维数组

在本小节中，将介绍如何在 MATLAB 中使用低维数组创建三维数组。

例 2.4 使用低维数组来创建高维数组。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> D2=[1,2,3;4,5,6;7,8,9];
>> D3(:,:,1)=D2;
>> D3(:,:,2)=2*D2;
>> D3(:,:,3)=3*D2;d
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>>D2
D2 =
     1      2      3
     4      5      6
     7      8      9
>> D3
D3(:,:,1) =
     1      2      3
     4      5      6
     7      8      9
D3(:,:,2) =
     2      4      6
     8     10     12
    14     16     18
D3(:,:,3) =
     3      6      9
    12     15     18
    21     24     27
```



从上面的结果中可以看出，由于三维数组中"包含"二维数组，因此可以通过二维数组来创建各种三维数组。

2.1.5 使用创建函数创建三维数组

在本小节中，将介绍如何利用 MATLAB 的创建函数来创建三维数组。

例 2.5 使用函数命令来创建高维数组。

step 1 使用 cat 命令来创建高维数组。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> D2=[1,2,3;4,5,6;7,8,9];
>> C=cat(3,D2,2*D2,3*D2);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
C(:,:,1) =
     1     2     3
     4     5     6
     7     8     9
C(:,:,2) =
     2     4     6
     8    10    12
    14    16    18
C(:,:,3) =
     3     6     9
    12    15    18
    21    24    27
```



cat 命令的功能是连接数组，其调用格式为 C = cat(dim,A1,A2,A3,A4...)其中，dim 表示创建数组的维度，A1,A2,A3,A4 表示的是各维度上的数组。

step 3 使用 repmat 命令来创建数组。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> D2=[1,2,3;4,5,6;7,8,9];
>> D3 = repmat(D2,2,3);
>> D4=repmat(D2,[1 2 3]);
```

step 4 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
D2 =
     1     2     3
     4     5     6
     7     8     9
D3 =
     1     2     3     1     2     3     1     2     3
     4     5     6     4     5     6     4     5     6
     7     8     9     7     8     9     7     8     9
     1     2     3     1     2     3     1     2     3
     4     5     6     4     5     6     4     5     6
     7     8     9     7     8     9     7     8     9
D4(:,:,1) =
     1     2     3     1     2     3
     4     5     6     4     5     6
     7     8     9     7     8     9
D4(:,:,2) =
     1     2     3     1     2     3
     4     5     6     4     5     6
     7     8     9     7     8     9
```

```
D4(:,:,3) =
     1     2     3     1     2     3
     4     5     6     4     5     6
     7     8     9     7     8     9
```



repmat 命令的功能在于复制并堆砌数组，其调用格式 B = repmat(A,[m n p...]) 中，A表示复制的数组模块，第二个输入参数则表示该数组模块在各个维度上的复制个数。

step 5 使用 reshape 命令来创建数组。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> D2=[1,2,3,4;5,6,7,8;9,10,11,12];
>> D3=reshape(D2,2,2,3);
>> D4=reshape(D2,2,3,2);
>> D5=reshape(D2,3,2,2)
```

step 6 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>> D2
D2 =
     1     2     3     4
     5     6     7     8
     9    10    11    12
>> D3
D3(:,:,1) =
     1     9
     5     2
D3(:,:,2) =
     6     3
    10     7
D3(:,:,3) =
    11     8
     4    12
>> D4
D4(:,:,1) =
     1     9     6
     5     2    10
D4(:,:,2) =
     3    11     8
     7     4    12
>> D5
D5(:,:,1) =
     1     2
     5     6
     9    10
D5(:,:,2) =
     3     4
     7     8
    11    12
```



reshape命令的功能在于修改数组的大小，因此用户可以将二维数组通过该命令修改为三维的数组，其调用格式为 B = reshape(A,[m n p ...])，其中A就是待重组的矩阵，后面的输入参数则表示数组各维的维度。

2.1.6 创建低维标准数组

除了前面小节中介绍的方法外，MATLAB 还提供多种函数来生成一些标准数组，用户可以直接使用这些命令来创建一些特殊的数组。在本小节中，将使用一些简单的例子来说明如何创建标准数组。

例 2.6 使用标准数组命令创建低维数组。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> A=zeros(3,2);
>> B=ones(2,4);
>> C=eye(4);
>> D=magic(5);
>> randn('state',0);
>> E=randn(1,2);
>> F=gallery(5);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
A =
     0     0
     0     0
     0     0
B =
     1     1     1     1
     1     1     1     1
C =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
D =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
E =
   -0.4326   -1.6656
F =
          -9           11          -21           63         -252
          70          -69          141         -421         1684
        -575          575        -1149         3451       -13801
        3891        -3891         7782       -23345        93365
        1024        -1024         2048        -6144        24572
```



并不是所有的标准函数命令都可以创建多种矩阵，例如 eye、magic 等命令就不能创建高维数组。同时，对于每个标准函数，参数都有各自的要求，例如 gallery 命令中只能选择 3 或者 5。

2.1.7 创建高维标准数组

在本小节中，将介绍如何使用标准数组函数来创建高维标准数组。

例 2.7 使用标准数组命令创建高维数组。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
% 设置随机数据器的初始条件
>> rand('state',1111);
>>D1=randn(2,3,5);
>> D2=ones(2,3,4);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>> D1
D1(:,:,1) =
    0.8156    1.2902    1.1908
    0.7119    0.6686   -1.2025
D1(:,:,2) =
   -0.0198   -1.6041   -1.0565
   -0.1567    0.2573    1.4151
D1(:,:,3) =
   -0.8051    0.2193   -2.1707
    0.5287   -0.9219   -0.0592
D1(:,:,4) =
   -1.0106    0.5077    0.5913
    0.6145    1.6924   -0.6436
D1(:,:,5) =
    0.3803   -0.0195    0.0000
   -1.0091   -0.0482   -0.3179
>> D2
D2(:,:,1) =
     1     1     1
     1     1     1
D2(:,:,2) =
     1     1     1
     1     1     1
D2(:,:,3) =
     1     1     1
     1     1     1
D2(:,:,4) =
     1     1     1
     1     1     1
```



限于篇幅，在这里就不细讲各种命令的详细参数和使用方法了，有需要的读者请自行阅读相应的帮助文件。

2.2 操作数值数组

在 MATLAB 中，除了需要创建数组之外，还需要对数组进行各种操作，包括重组、元素变换、提取和旋转等操作，MATLAB 都提供了对应的函数命令，在本节中将以简单的实例来说明这些命令的使用方法。由于数组维度的不同将会带来 MALTAB 不同的操作要求，因此将分数组维度的差别来讨论数组的操作。

2.2.1 选取低维数组的对角元素

和前面介绍的相同，低维数组的操作比高维数组的操作在运算或者使用上要简单，因此在本小节中将首先使用实例来介绍如何操作低维数组。

例 2.8 使用 diag 命令来选取对角元素或者创建矩阵。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> Data=[1,2,3,4;5,6,7,8;9,10,11,12];
A1=diag(Data,1);A2=diag(Data);
A3=diag(Data,-1);
B1=diag(A2,1);
B2=diag(A3,1);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
Data =
     1     2     3     4
     5     6     7     8
     9    10    11    12
A1 =
     2
     7
    12
A2 =
     1
     6
    11
A3 =
     5
    10
B1 =
     0     1     0     0
     0     0     6     0
     0     0     0    11
     0     0     0     0
B2 =
     0     5     0
     0     0    10
     0     0     0
```



从上面的结果中可以看出，diag 命令的功能可以是选取矩阵对角线的数组，也可以将某个数组来创建矩阵，可以很方便地利用该命令处理矩阵对角线的数据。

对于 diag 命令中参数 "k" 的含义，可以用图 2.1 来形象地说明。

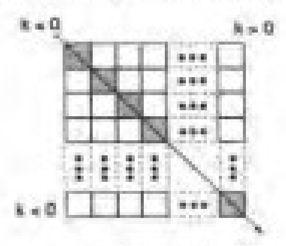


图 2.1 diag 命令中参数 k 的含义



低维数组的形式转换

例 2.9 对数组或者矩阵来进行形式转换：对称变换和旋转。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> Data=[1,2,3,4;5,6,7,8;9,10,11,12];
%矩阵的转置
>> B=Data';
>> C=fliplr(Data);
>> D=flipud(Data);
%多次旋转矩阵
>> E=rot90(Data);
>> F=rot90(E);
>> G=rot90(F);
>> H=rot90(G);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
Data =
     1     2     3     4
     5     6     7     8
     9    10    11    12
B =
     1     5     9
     2     6    10
     3     7    11
     4     8    12
C =
     4     3     2     1
     8     7     6     5
    12    11    10     9
D =
     9    10    11    12
     5     6     7     8
     1     2     3     4
E =
     4     8    12
     3     7    11
     2     6    10
     1     5     9
F =
    12    11    10     9
     8     7     6     5
     4     3     2     1
G =
     9     5     1
    10     6     2
    11     7     3
    12     8     4
H =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

上面的程序代码中演示了各种转换命令，下面简要介绍各种命令的含义。

- fliplr：以数组的垂直中线为对称轴，交换左右对称位置上的数组元素。
- flipud：以数组的水平中线为对称轴，交换左右上下对称位置上的数组元素。
- rot90：逆时针旋转二维数组 90° 。

选取三角矩阵

例 2.10 选取数组上三角或者下三角矩阵。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> Data=[ 1,2,3,4;5,6,7,8;9,10,11,12] ;
>> Al=tril(Data);
>> Bl=tril(Data,1);
>> Cl=tril(Data,2);
>> Dl=tril(Data,-1);
>> Dl=tril(Data,-2);
>> Au=triu(Data);
>>Bu=triu(Data,1);
>>Cu=triu(Data,2);
>>Du=triu(Data,-1);
>>Eu=triu(Data,-2);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
Data =
     1     2     3     4
     5     6     7     8
     9    10    11    12
Al =
     1     0     0     0
     5     6     0     0
     9    10    11     0
Bl =
     1     2     0     0
     5     6     7     0
     9    10    11    12
Cl =
     1     2     3     0
     5     6     7     8
     9    10    11    12
Dl =
     0     0     0     0
     5     0     0     0
     9    10     0     0
El =
     0     0     0     0
     0     0     0     0
     9     0     0     0
Au =
     1     2     3     4
     0     6     7     8
     0     0    11    12
```

```
Bu =
     0     2     3     4
     0     0     7     8
     0     0     0    12
Cu =
     0     0     3     4
     0     0     0     8
     0     0     0     0
Du =
     1     2     3     4
     5     6     7     8
     0    10    11    12
Eu =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```



从上面的步骤中，tril命令的功能是提取矩阵的下三角矩阵，triu命令的功能在于提取矩阵的上三角矩阵，两个命令中的参数k的含义和diag命令相同。

2.2.4 Kronecker 乘法

例 2.11 演示Kronecker乘法的不可交换性。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>> X=[1,2;3,4];
>>I=eye(3);
>> A=kron(X,I);
>> B=kron(I,X);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
X =
     1     2
     3     4
I =
     1     0     0
     0     1     0
     0     0     1
A =
     1     0     0     2     0     0
     0     1     0     0     2     0
     0     0     1     0     0     2
     3     0     0     4     0     0
     0     3     0     0     4     0
     0     0     3     0     0     4
B =
     1     2     0     0     0     0
     3     4     0     0     0     0
     0     0     1     2     0     0
     0     0     3     4     0     0
     0     0     0     0     1     2
     0     0     0     0     3     4
```



从上面的结果中可以看出，对于相同的两个矩阵，交换后的乘积是不同的结果，因此就证明了Kronecker乘法的不可交换性，关于Kronecker乘法，请读者自行查阅相应的书籍。

2.2.5 高维数组的对称交换

对于高维数组，由于在结构上多了维度，因此在操作方法上多了一些操作其他维度的命令，在本小节中还是以简单的实例来介绍这些函数的使用方法。

例 2.12 对三维数组进行对称交换。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> Data=[1,2,3,4;5,6,7,8;9,10,11,12];
>> A=reshape(Data,2,2,3);
>> B=flipdim(A,1);
>> C=flipdim(A,2);
>> D=flipdim(A,3);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>> A
A(:,:,1) =
     1     9
     5     2
A(:,:,2) =
     6     3
    10     7
A(:,:,3) =
    11     8
     4    12
>> B
B(:,:,1) =
     5     2
     1     9
B(:,:,2) =
    10     7
     6     3
B(:,:,3) =
     4    12
    11     8
>> C
C(:,:,1) =
     9     1
     2     5
C(:,:,2) =
     3     6
     7    10
C(:,:,3) =
     8    11
    12     4
>> D
D(:,:,1) =
```

```
    11     8
     4    12
D(:,:,2) =
     6     3
    10     7
D(:,:,3) =
     1     9
     5     2
```



在MATLAB中，flipdim(A,k)命令中第一个输入变量A表示的是被操作的数组，第二个输入变量k指定的是对称面。1表示的是与数组行平行的平分面，2表示的是与数据列平行的平分面，3表示的是与数据页平行的平分面。

2.2.6 高维数组的维序号移动

例2.13 对三维数组进行"维序号移动"。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>> Data=[1,2,3,4;5,6,7,8;9,10,11,12];
>> A=reshape(Data,2,2,3);
>> Adim=shiftdim(A,1);
>> Adim2=shiftdim(A,2);
>> Adim3=shiftdim(A,3);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>> A
A(:,:,1) =
     1     9
     5     2
A(:,:,2) =
     6     3
    10     7
A(:,:,3) =
    11     8
     4    12
>> Adim
Adim(:,:,1) =
     1     6    11
     9     3     8
Adim(:,:,2) =
     5    10     4
     2     7    12
>> Adim2
Adim2(:,:,1) =
     1     5
     6    10
    11     4
Adim2(:,:,2) =
     9     2
     3     7
     8    12
```

```
>> Adim3
Adim3(:,:,1) =
      1     9
      5     2
Adim3(:,:,2) =
      6     3
     10     7
Adim3(:,:,3) =
     11     8
      4    12
```

高维数组的广义共轭转置

例 2.14 对三维数组进行广义共轭转置。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> A=reshape(Data,2,2,3);
>> Ap1=permute(A,[ 1,2,3] );
>> Ap2=permute(A,[ 2,3,1] );
>> Ap3=permute(A,[ 3,2,1] );
>> Ap4=permute(A,[ 3,1,2] );
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>> A
A(:,:,1) =
      1     9
      5     2
A(:,:,2) =
      6     3
     10     7
A(:,:,3) =
     11     8
      4    12
>> Ap1
Ap1(:,:,1) =
      1     9
      5     2
Ap1(:,:,2) =
      6     3
     10     7
Ap1(:,:,3) =
     11     8
      4    12
>> Ap2
Ap2(:,:,1) =
      1     6    11
      9     3     8
Ap2(:,:,2) =
      5    10     4
      2     7    12
>> Ap3
Ap3(:,:,1) =
      1     9
```

```
      6     3
     11     8
Ap3(:,:,2) =
      5     2
     10     7
      4    12
>> Ap4
Ap4(:,:,1) =
      1     5
      6    10
     11     4
Ap4(:,:,2) =
      9     2
      3     7
      8    12
```



在 MATLAB 中，permute 命令的第一个输入变量是被转置的数组，第二个变量表示的是转置方式的行数组，该行数组中元素位置号代表的是新数组的维序号数值。

2.2.8 高维数组的降维操作

例 2.15 使用 squeeze 命令来撤销“孤维”，使高维数组进行降维。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> B=cat(4,A(:,:,1),A(:,:,2),A(:,:,3));
>> C=squeeze(B);
>> size_B=size(B);
>> size_C=size(C);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>> B
B(:,:,1,1) =
      1     9
      5     2
B(:,:,1,2) =
      6     3
     10     7
B(:,:,1,3) =
     11     8
      4    12
>> C
C(:,:,1) =
      1     9
      5     2
C(:,:,2) =
      6     3
     10     7
C(:,:,3) =
     11     8
      4    12
>> size_B
```

```
size_B =
     2     2     1     3
>> size_C
size_C =
     2     2     3
```



稀疏矩阵

在MATLAB中，系统一般使用两种方法来存储数据，也就是满矩阵的形式和稀疏矩阵的形式，简称满矩阵和稀疏矩阵。在很多情况下一个矩阵中只有少数元素是非零的，对于满矩阵MATLAB会使用相同的空间来储存零元素和非零元素，这种储存方法对于大多数元素为0的稀疏矩阵而言，将会造成大量的浪费。因此，对于稀疏矩阵，MATLAB提供特殊的存储方法，同时提供特殊的操作函数和运算法则，下面详细介绍。

稀疏矩阵的存储方式

在MATLAB中，一般使用3个矩阵来存储稀疏矩阵，假设有一个$m \times n$的矩阵，其中有*nnz*个非零元素，存储在长度为*nz* max的矩阵中。

- 第一个矩阵用来存储所有的非零元素，该矩阵的长度为*nz* max。
- 第二个矩阵用来存储所有的非零元素的行指标，该矩阵的长度也是*nz* max。
- 第三个矩阵用来存储每一列的开始处指针和一个标志着这3个矩阵结束的指针，该矩阵的长度为n+1。

根据上面的介绍，一个稀疏矩阵需要存储*nz* max个浮点数和*nz* max+n+1个整数，因此，存储一个稀疏矩阵需要8* *nz* max+4*(*nz* max+n+1)个字节的单元。

对于稀疏矩阵和满矩阵的存储差异，MATLAB提供下面的转换命令：

- SM = Sparse (A) 将其他存储方式转换为其他的稀疏矩阵形式；
- FM = Full (A) 把矩阵存储方式从任何一个存储形式转换为满矩阵形式。

2.3.2　使用sparse命令创建稀疏矩阵

由于满矩阵的运算得到的结果还是满矩阵，因此如果不通过相应的命令将不会创建稀疏矩阵。在MATLAB中，提供多个命令来创建稀疏矩阵，经常使用的有sparse和spdiags两种，对应的调用格式如下：

S = sparse(i,j,s,m,n,nzmax)　使用[I,j,s] 的行创建$m \times n$维稀疏矩阵S；

- 在上面的命令中，s表示的是按照排列的所有非零元素构成的向量。i，j分别表示非零元素的行下标和列下标向量。

A = spdiags(B,d,m,n)　抽取和创建带、对角稀疏矩阵

- 在上面的参数中，m、n分别表示指定矩阵的行和列的维数。d表示的是长度为p的整数向量，B是满矩阵，用来指定A矩阵的对角线位置上的元素。



在 MATLAB 中，还提供从外部数据转换为稀疏矩阵的命令 spconvert，用户可以首先通过 load 命令加载外部数据，然后使用该命令创建稀疏矩阵。

例 2.16 使用 sparse 命令创建稀疏矩阵。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
A = [ 0   0   0   5
      0   2   0   0
      1   3   0   0
      0   0   4   0];
S1 = sparse(A);
B=full(S1);
n=5;
D = sparse(1:n,1:n,-2*ones(1,n),n,n);
E = sparse(2:n,1:n-1,ones(1,n-1),n,n);
S2= E+D+E';
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
S1 =
   (3,1)        1
   (2,2)        2
   (3,2)        3
   (4,3)        4
   (1,4)        5
B =
     0     0     0     5
     0     2     0     0
     1     3     0     0
     0     0     4     0
D =
   (1,1)       -2
   (2,2)       -2
   (3,3)       -2
   (4,4)       -2
   (5,5)       -2
E =
   (2,1)        1
   (3,2)        1
   (4,3)        1
   (5,4)        1
S2 =
   (1,1)       -2
   (2,1)        1
   (1,2)        1
   (2,2)       -2
   (3,2)        1
   (2,3)        1
   (3,3)       -2
   (4,3)        1
   (3,4)        1
   (4,4)       -2
   (5,4)        1
   (4,5)        1
```

```
   (5,5)       -2
```

step 3 查看存储信息。为了加强用户对稀疏矩阵的存储信息的理解，可以使用whos命令查看各变量的信息：

```
>> whos
  Name      Size                    Bytes  Class
  A         4x4                       128  double array
  B         4x4                       128  double array
  D         5x5                        84  double array (sparse)
  E         5x5                        72  double array (sparse)
  S1        4x4                        80  double array (sparse)
  S2        5x5                       180  double array (sparse)
  n         1x1                         8  double array
Grand total is 60 elements using 680 bytes
```

使用spdiags命令创建稀疏矩阵

例2.17 使用spdiags命令来创建稀疏矩阵。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>> B = [ 41     11      0
         52     22      0
         63     33     13
         74     44     24 ];
>> d = [-3
         0
         2];
>> S=spdiags(B,d,7,4);
>> D=full(S);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
S =
   (1,1)        11
   (4,1)        41
   (2,2)        22
   (5,2)        52
   (1,3)        13
   (3,3)        33
   (6,3)        63
   (2,4)        24
   (4,4)        44
   (7,4)        74
D =
    11     0    13     0
     0    22     0    24
     0     0    33     0
    41     0     0    44
     0    52     0     0
     0     0    63     0
     0     0     0    74
```



在MATLAB中，还提供创建特殊稀疏矩阵的各种命令，例如speye、spones、sprand、sprandsym等。关于这些命令的详细使用方法，用户可以查看相应的帮助文件。

2.3.4 查看稀疏矩阵的信息

由于稀疏矩阵的维度一般比较大，直接查看稀疏矩阵不利于用户查看稀疏矩阵的信息。为此，MATLAB提供查看稀疏矩阵定量信息和图形化信息的函数，主要用来查看稀疏矩阵的非零元素信息和图形化稀疏矩阵信息。

- n=nnz(x) 查看稀疏矩阵中的非零元素个数。
- s=nonzeros(A) 返回稀疏矩阵中的非零元素数值。
- n=nzmax(S) 返回稀疏矩阵中存储非零元素的空间长度。

例 2.18 查看某稀疏矩阵的元素信息。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>> load west0479
>> S=west0479;
>> n1=nnz(S);
>> s1=nonzeros(S);
>> n2=nzmax(S);
>> format short e
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
n1 =
        1887
n2 =
        1887
s1 =
 1.0000e+000
 -3.7648e-002
 -3.4424e-001
  1.0000e+000
 -2.4523e-002
 -3.7371e-001
  1.0000e+000
 -3.6613e-002
 ......................// 省略了部分数据
  3.6044e-001
  2.0539e-002
  7.1093e-001
  3.1305e+000
  1.3574e-001
  1.0000e+000
  2.8831e-001
  7.1490e-002
```



从上面的结果中可以看出，在默认的情况下，稀疏矩阵中非零元素的个数和存储非零元素的长度相同。

2.3.5 稀疏矩阵的图形化信息

除了上面的信息查询函数之外，MATLAB提供查看稀疏矩阵的图形化命令spy，其具体的调用格式为：

spy(S,'LineSpec',markersize) 其中S表示的是稀疏矩阵，LineSpec表示的是线型属性的字符串，markersize则表示标记大小。

例2.19 查看稀疏矩阵的图形化信息。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>> load west0479
>>S=west0479;
>> spy(S)
```

step 2 在输入上面的程序代码后，按"Enter"键，得到的图形如图2.2所示。

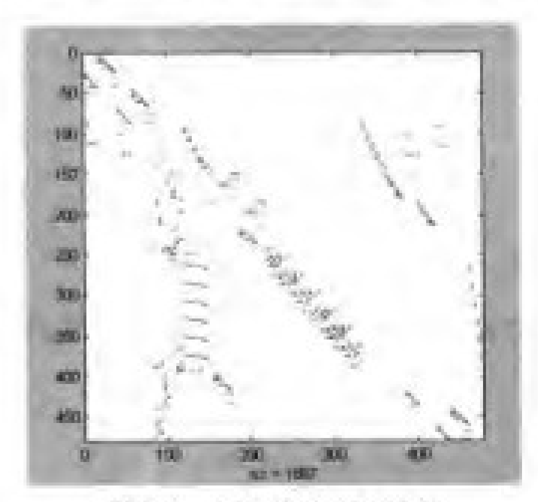


图2.2 稀疏矩阵S的图形

step 3 在MATLAB的命令窗口中输入下面的程序代码：

```
>> B=bucky;
>> C=B^2;
>> D=B^4;
>> E=B^6;
>>subplot(221);
>>spy(B)
>>subplot(222);
>>spy(C)
>>subplot(223);
>>spy(D)
>>subplot(224);
>>spy(E)
```

step 4 在输入上面的程序代码后，按"Enter"键，得到的图形如图2.3所示。

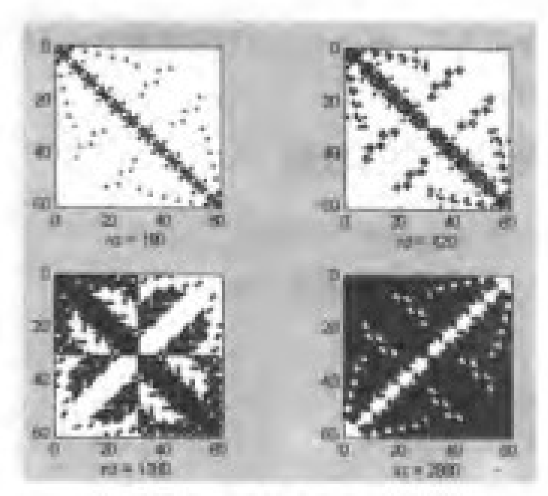


图 2.3 稀疏矩阵的图形

2.4 字符串数组

尽管MATLAB的主要运算对象是数值数组，但是在实际运用中也会经常需要处理字符串对象，因此在MATLAB中也提供字符串数组相关操作的函数。正是因为字符串和数值属于不同的数据类型，所以在创建和操作中有许多地方和数值数组有明显的差异，本节将分别介绍字符串数组的相关操作情况。

2.4.1 直接输入法创建字符串数组

在MATLAB中，用户可以使用多种方法来创建字符串数组，本小节将以各种简单的实例来说明如何创建各种要求的字符串数组。

例 2.20 通过直接输入法来创建字符串数组。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
a='Char';
b='字符串数组';
c='Example''2,1''';
d=[b,' ',c,'.'];
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
a =
Char
b =
字符串数组
c =
Example'2.1'
d =
字符串数组 Example'2.1'.
```

从上面的程序代码可以得知创建字符串数组的基本方法：

- 直接在单引号对内输入字符串的内容；

- 当字符串文字中包含单引号时，每个单引号符号需要使用连续的 2 个单引号字符；
- 可以使用小的字符串构成长的字符串。

2.4.2 使用 ASCII 码创建字符串数组

例 2.21 通过 ASCII 码的转换来创建字符串数组。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> b='字符串数组';
>> ascii_b=double(b);
>> c=char(ascii_b);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
b =
字符串数组
ascii_b =
      23383        31526        20018        25968        32452
c =
字符串数组
```



从上面的结果可以看出，在 MATLAB 中，用户可以使用 char 和 double 在字符串数组和数值数组之间转换，其中数值数组就是字符串对应的 ASCII 码。

step 3 通过 ASCII 码实现字符串变量大小写的转换。在命令窗口中输入下面的程序代码：

```
>> charA='Matlab 7.0 String Data';
>>w=find(charA>='a'&charA<='z');
>>asciiA=double(charA);
>>asciiA(w)=asciiA(w)-32;
>> charB=char(asciiA);
```

step 4 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
charA =
Matlab 7.0 String Data
charB =
MATLAB 7.0 STRING DATA
```



在 MATLAB 中，大小写字符串的 ASCII 码相差 32，因此使用数值数组之间的加法或者减法来实现大小写变换。

2.4.3 使用函数创建字符串数组

例 2.22 通过数组创建函数来创建字符串数组。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> ch1=char('Matlab 7.0','String Data');
```

```
>> ch2=str2mat('Matlab','7.0','','String','Data');
>> t1 = 'first';
>> t2 = 'string';
>> t3= 'matrix';
>> t4 = 'second';
>> S1 = strvcat(t1, t2, t3);
>> S2 = strvcat(t4, t2, t3);
>> S3 = strvcat(S1, S2);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
ch1 =
Matlab 7.0
String Data

ch2 =
Matlab
7.0

String
Data

S1 =
first
string
matrix

S2 =

second
string
matrix

S3 =

first
string
matrix
second
string
matrix
```



在MATLAB中，使用上面三个函数来创建字符串时，不需要注意每个字符串变量的长度是否相等，同时用户应该注意空串和空格串的差别。

2.4.4 处理字符串数组的空格

为了方便用户操作字符串数组，MATLAB中提供多种字符串的操作函数，包括对字符串进行转换、裁减、连接、查找等功能。这些函数内容繁多，在本章中将主要介绍比较常见的字符串操作函数，其他具体函数请用户自行查阅帮助文件。

例 2.23 使用不同的方法来处理字符串中的空格。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>> A{1,1} = 'MATLAB    ';
>> A{1,2} = 'SIMULINK    ';
>> A{2,1} = 'Toolboxes    ';
>> A{2,2} = 'The MathWorks    ';
>> B=deblank(A);
>> cstr = {'     Trim leading white-space';
'Trim trailing white-space     '};
>>cstrim=strtrim(cstr);
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
A =
    'MATLAB    '       'SIMULINK    '
    'Toolboxes    '    'The MathWorks    '
B =
    'MATLAB'       'SIMULINK'
    'Toolboxes'    'The MathWorks'
cstr =
    '     Trim leading white-space'
    'Trim trailing white-space     '
cstrim =
    'Trim leading white-space'
    'Trim trailing white-space'
```



从上面的结果可以看出，两个不同的命令都可以清除字符串中的空格，但是具体的用法和功能略有不同，strtrim命令的功能主要是清除字符串前后的空格，deblank命令的功能则是清除字符串尾部的空格。

2.4.5 读取字符串数组的信息

例 2.24 使用函数来查找、替代和读取字符串中的信息。

step 1 替代字符串。在MATLAB的命令窗口中输入下面的程序代码：

```
>> s1 = 'This is a good example.';
>>str = strrep(s1, 'good', 'great');
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
s1 =
This is a good example.
str =
This is a great example.
```



在上面的结果中，字符串变量str通过命令strrep将字符串变量s1中的“good”替换为“great”。

step 3 读取字符串数组的信息。在MATLAB的命令窗口中输入下面的程序代码：

```
>> str = '<table border=5 width="100%" cellspacing=0>';
```

```
[border width space] = strread(str, ...
    '%*s%*s %c %*s "%4s" %*s %c', 'delimiter', '= ');
```

step 4 查看程序结果。按"Enter"键，可以得到下面的程序结果：

```
border =
5
width =
    '100%'
space =
0
```



在上面的程序代码中，通过使用strread从原始的字符串中读取各种具体有效的数值信息。

step 5 查找字符串中的"< >"符号，得出有效信息。在MATLAB的命令窗口中输入下面的程序代码：

```
>> s = sprintf('%s%s%s', ...
'<ul class=continued><li class=continued><pre>', ...
'<a name="13474"></a>token = strtok(''str'', delimiter)', ...
'<a name="13475"></a>token = strtok(''str'')');
rem = s;
for k=1:11
   [t{k}, rem] = strtok(rem, '<>');
   if isempty(t{k}),  break;  end
   disp(sprintf('%s',t{k}))
   end
```

step 6 查看程序结果。按"Enter"键，可以得到下面的程序结果：

```
ul class=continued
li class=continued
pre
a name="13474"
/a
token = strtok('str', delimiter)
a name="13475"
/a
token = strtok('str')
```



在上面的程序代码中，首先将某个HTML代码中的字符串添加到变量s中，然后将这个字符串切断为各个部分，并删除"<"和">"字符串，最后使用循环语句来将各个字符串信息显示出来。

2.5 构架数组

在MATLAB中，构架数组是一个比较特殊的数组类型，在该数组中可以存放各种不同类型的数据。构架数组的基本组成是构架，数组中的每个构架都是平等的，它们主要用下标来区分。构架必须在划

分“域”之后才能使用，数据不能直接存放在构架数组中，而只能直接存放在构架的域中。而构架中的“域”可以存放任何类型、任何大小的数组，而且不同构架数组中的同名域中可以存放不同的内容。

鉴于构架数组具有上面这么多的特殊之处，本节将介绍如何在MATLAB中创建、操作和运用构架数组。



使用直接法创建单构架数组

构架数组实质上具有面向对象中的数据结构功能，具有属性名称、属性数值两个主要对象域，对此MATLAB提供多种创建方法来构建构架数组。在本小节中，将以具体的实例来说明如何创建构架数组。

例 2.25 使用直接法来创建某个关于病人的构架数组。

step 1 创建维度为 1 × 1 的构架数组。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> patient.name = 'John Doe';
>>patient.billing = 127.00;
>>patient.test = [ 79 75 73; 180 178 177.5; 220 210 205];
>> patient.medi_information.city='NewYork';
>> patient.medi_information.date='2006/7/1';
```

step 2 查看该构架数组的存储信息。使用 whos 命令查看数组信息，其具体的信息如下：

```
>> whos patient
  Name          Size                    Bytes  Class
  patient       1x1                       870  struct array

Grand total is 39 elements using 870 bytes
```

step 2 查看该构架数组的具体信息。在 MATLAB 的命令窗口中输入下面的代码：

```
>> patient
patient =
               name: 'John Doe'
            billing: 127
               test: [ 3x3 double]
    medi_information: [ 1x1 struct]

>> patient.medi_information
ans =
    city: 'NewYork'
    date: '2006/7/1'
>> patient.test
ans =
   79.0000   75.0000   73.0000
  180.0000  178.0000  177.5000
  220.0000  210.0000  205.0000
>> patient.medi_information.city
ans =
NewYork
```

在上面的程序代码中，patient 只有一个构架，其总共有 4 个数值域：name、billing、test 和

medi_information。其中medi_information又有city和date两个子域。向子域中直接添加数值是创建构架数组的最常用方法。



当用户在命令窗口中直接输入构架名称的时候，MATLAB只会显示构架的域名以及对应的信息；如果需要查看各个构架数组中具体名域的数值时，则需要引用对应的具体名称。

2.5.2 使用直接法创建二维构架数组

例 2.26 沿用上面的实例，创建二维构架数组。

step 1 直接扩充构架数组的维度。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>>patient(2).name = 'Ann Lane';
>>patient(2).billing = 28.50;
>>patient(2).test = [ 68 70 68; 118 118 119; 172 170 169];
>>patient(2).medi_information.city='Chicago';
>>patient(2).medi_information.date='2006/8/1';
>>patient(3).name = 'Ann Smitch';
>>patient(3).billing = 504.70;
>>patient(3).test = [ 80 80 68; 153 153 154; 181 190 182];
>>patient(3).medi_information.city='Boston';
>>patient(3).medi_information.date='2006/9/1';
```

step 2 查看该构架数组的存储信息。使用 whos 命令查看数组信息，其具体的信息如下：

```
>> whos patient
  Name          Size                    Bytes  Class
  patient       1x3                      2100  struct array

Grand total is 118 elements using 2100 bytes
```

step 3 利用上面的数组创建新数组。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>>mypatient=patient(1:2);
```

step 4 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>> mypatient
mypatient =

1x2 struct array with fields:
    name
    billing
    test
    medi_information
```

step 5 访问构架数组的信息。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> mypatient(1)
ans =
```

```
              name: 'John Doe'
           billing: 127
              test: [ 3x3 double]
   medi_information: [ 1x1 struct]
>> name=mypatient(2).name
name =
Ann Lane
>> test=mypatient(2).test
test =

    68    70    68
   118   118   119
   172   170   169
```



从上面的结果中可以看出，对于高维构架数组，用户也可以使用直接对子域进行赋值来创建，同时可以使用高维构架数组来创建低维构架数组。

2.5.3 使用直接法创建三维构架数组

例 2.27 沿用上面的实例，创建三维构架数组。

step 1 直接扩充构架数组的维度。在 MATLAB 的命令窗口中输入下面的程序代码：

```
patient(1,1,1).name = 'John Doe';patient(1,1,1).billing = 127.00;
patient(1,1,1).test =[ 79 75 73; 180 178 177.5; 220 210 205];
patient(1,2,1).name = 'Ann Lane';patient(1,2,1).billing = 28.50;
patient(1,2,1).test =[ 68 70 68; 118 118 119; 172 170 169];
patient(1,1,2).name = 'Al Smith';patient(1,1,2).billing = 504.70;
patient(1,1,2).test =[ 80 80 80; 153 153 154; 181 190 182];
patient(1,2,2).name = 'Dora Jones';patient(1,2,2).billing = 1173.90;
patient(1,2,2).test =[ 73 73 75; 103 103 102; 201 198 200];
```

step 2 查看该构架数组的存储信息。使用 whos 命令查看数组信息，其具体的信息如下：

```
>> patient
patient =

1x3x2 struct array with fields:
    name
    billing
    test
    medi_information
```

step 3 查看该构架数组的结构图。由于三维构架数组比较难以理解，下面的图形列出了具体的结构图，如图 2.4 所示。

2.5.4 使用命令创建构架数组

例 2.28 使用 struct 命令创建构架数组。

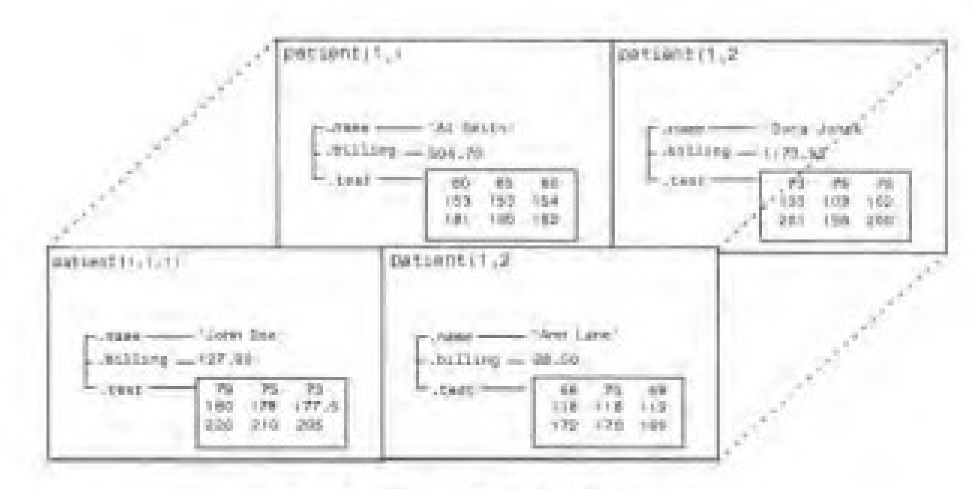


图 2.4 三维构架数组的图形化结构

step 1 调用 struct 命令，创建构架数组。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>>W = struct('city', {'Bos','Chi','Lin','Dnv','Vgs','SFr'}, ...
             'temp', {43, 34, 25, 15, 31, 52}, ...
             'heatix', {32, 27, 17, -5, 22, 47}, ...
             'wspeed', {8, 3, 11, 9, 4, 18}, ...
             'wchill', {37, 30, 16, 0, 35, 42});
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>> W
W =

1x6 struct array with fields:
    city
    temp
    heatix
    wspeed
    wchill
```



在 MATLAB 中，创建构架数组的命令 struct 的调用格式为 s = struct('field1', values1, 'field2', values2, ...)，其中 field 表示构架数组的域，values1 表示数组域的数值。

2.5.5 访问构架数组的数据

由于构架数组比较特殊，如果用户希望在其他的应用程序或者模块中使用构架数组，则需要访问构架数组中的数据，下面使用例子来说明如何访问构架数组的数据。

例 2.29 访问构架数组的数据。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> my1=mypatient(1);
>> test2b = patient(3).test(2,2);
>> bills = [patient.billing];
>> tests = {patient(1:2).test};
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
>> my1
my1 =

                 name: 'John Doe'
              billing: 127
                 test: [ 3x3 double]
    medi_information: [ 1x1 struct]

test2b =
   153

bills =
  127.0000   28.5000  504.7000
tests =

    [ 3x3 double]    [ 3x3 double]
```



在上面的程序代码中，patient 和 mypatient 变量是前面小节中创建的构架数组，上面的步骤中演示了不同的引用方法。

例 2.30 在程序代码中调用构架数组中的数据，并绘制对应的图形。

step 1 创建构架数组。在 MATLAB 的命令窗口中输入下面的程序代码：

```
test(1).lead = .007;
test(2).lead = .031;
test(3).lead = .019;
test(1).mercury = .0021;
test(2).mercury = .0009;
test(3).mercury = .0013;
test(1).chromium = .025;
test(2).chromium = .017;
test(3).chromium = .1
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
test =

1x3 struct array with fields:
    lead
    mercury
    chromium
```

step 3 编写绘图函数命令。选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function [ r1, r2] = concen(toxtest);
% 计算变量之间的相对比例
r1 = [ toxtest.mercury] ./ [ toxtest.lead] ;
r2 = [ toxtest.lead] ./ [ toxtest.chromium] ;
% 计算变量数值
```

```
lead = [ toxtest.lead] ;
mercury = [ toxtest.mercury] ;
chromium = [ toxtest.chromium] ;
%绘制三个变量的图形
plot(lead, 'r','LineWidth',1.5);    hold on
plot(mercury, 'b','LineWidth',1.5)
plot(chromium, 'y','LineWidth',1.5);    hold off
%添加图形图例和标题
legend('lead','mercury','chromium')
title('The ratio of mix')
grid
```

在输入上面的程序代码后，将上面的程序代码保存为 "concen.m" 文件。

step 4 运行函数。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> [ a,b] =concen(test)
```

step 5 查看结果。输入上面的程序代码后，得到的计算结果如下：

```
a =
    0.3000    0.0290    0.0684
b =
    0.2800    1.8235    0.1900
```

同时，可以得到的图形如图 2.5 所示。

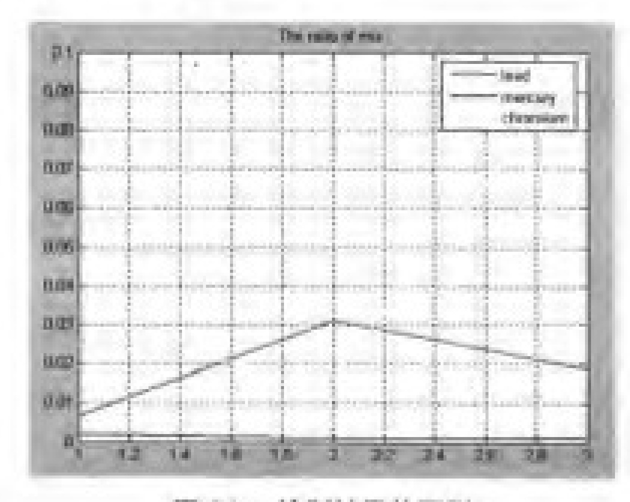


图 2.5　绘制结果的图形

2.5.6 设置构架数组的域属性

由于构架数组的域是存放数据的场所，因此调取和设置构架数组中数据的前提就是了解或者熟悉构架数组的域名。尽管在前面曾经介绍过访问构架数组域名的方法，但是这种方法并不能获得其他命令能够处理的域名。为了解决这个矛盾，MATLAB 提供关于处理域名的各种命令，在本小节中将主要介绍一些常见命令。

- names = fieldnames(s)：获取构架数组的域名
- f = getfield(s,'field')：获取具体构架数组中的内容

✦ s = setfield(s, 'field', v) 设置具体构架数组中的内容

例 2.31 使用 fieldnames 命令获取构架数组的域属性。

step 1 创建构架数组。在 MATLAB 的命令窗口中输入下面的程序代码：

```
mystr(1,1).name = 'alice';
mystr(1,1).ID = 0;
mystr(2,1).name = 'gertrude';
mystr(2,1).ID = 1
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
mystr =
2x1 struct array with fields:
    name
    ID
```

step 1 获取域名属性。在 MATLAB 的命令窗口中输入下面的程序代码：

```
n = fieldnames(mystr)
```

step 1 查看程序结果。按 "Enter" 键，可以得到下面的程序结果：

```
n =
    'name'
    'ID'
```

step 5 获取其他对象的域名属性。在 MATLAB 的命令窗口中输入下面的程序代码：

```
f = java.awt.Frame;
fieldnames(f)
```

step 6 查看程序结果。按 "Enter" 键，可以得到下面的程序结果：

```
ans =
    'DEFAULT_CURSOR'
    'CROSSHAIR_CURSOR'
    'TEXT_CURSOR'
    'WAIT_CURSOR'
    ....................................// 省略了部分数据
    'BOTTOM_ALIGNMENT'
    'LEFT_ALIGNMENT'
    'RIGHT_ALIGNMENT'
    'WIDTH'
    'HEIGHT'
    'PROPERTIES'
    'SOMEBITS'
    'FRAMEBITS'
    'ALLBITS'
    'ERROR'
    'ABORT'
```

例 2.32 使用 getfield 命令来获取域属性。

step 1 创建构架数组。在 MATLAB 的命令窗口中输入下面的程序代码：

```
mystr(1,1).name = 'alice';
mystr(1,1).ID = 0;
mystr(2,1).name = 'gertrude';
mystr(2,1).ID = 1
```

step 2 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
mystr =
2x1 struct array with fields:
    name
    ID
```

step 1 获取域名属性。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> for k = 1:2
    name{ k} = getfield(mystr,{ k,1} ,'name');
end
```

step 4 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
name =

    'alice'    'gertrude'
```

例 2.33 使用 setfield 命令来设置域属性。

step 1 创建构架数组。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> class = 5;   student = 'John_Doe';
grades_Doe = [ 85, 89, 76, 93, 85, 91, 68, 84, 95, 73] ;
grades = [ ] ;
grades = setfield(grades, {class} , student, 'Math', ...
   { 10, 21:30} , grades_Doe);
```

step 4 查看程序结果。在命令窗口输入变量名称，可以得到下面的程序结果：

```
grades =

1x5 struct array with fields:
    John_Doe
```

step 3 查看构架数组的信息。在 MATLAB 的命令窗口中输入下面的程序代码：

```
grades(class).John_Doe.Math(10, 21:30)
```

step 1 查看程序结果。按 "Enter" 键，可以得到下面的程序结果：

```
ans =

85    89    76    93    85    91    68    84    95    73
```



小结

在本章中，依次介绍了如何在MATLAB中创建和操作数值数组、稀疏矩阵、字符串数组和构架数组。这些内容都是MATLAB中的基础知识，因此希望用户能够熟练掌握。其中数值数组是MATLAB所有操作的重要内容，字符串和构架数组在实际工程中也有广泛的应用。在后面的章节中，将主要介绍如何使用MATLAB进行数值运算。

◆ ◆ ◆

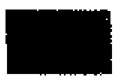
第3章 数值运算

本章包括

- ◆ 矩阵分析
- ◆ 矩阵分解
- ◆ 概率论和数理分析
- ◆ 线性方程组
- ◆ 数值积分

在前面已经介绍过，MATLAB的基本运算单元是数组，因此本章将从矩阵分析、线性代数的数值计算开始介绍，然后介绍函数的零点、数值积分、数理统计和分析等。在MATLAB中，数值运算主要通过函数或者命令来实现。由于在MATLAB中所有的数据都以矩阵的形式出现，其对应的数值运算包括两种类型：一种是针对整个矩阵的数值运算，也就是矩阵运算，例如求解矩阵行列式的函数det；另外一种是针对矩阵中的元素进行运算的函数，可以称为矩阵元素的运算，例如求解矩阵中每个元素的余弦函数cos等。

从整体的角度来看，本章的内容之间没有太大的递承关系，用户可以根据需要来选择阅读的章节。同时，为了方便大家阅读各自章节的内容，每个小节中的例子都是完整的，用户可以在自己的机器上运行。

 矩阵分析

矩阵分析是线性代数的重要内容，也是几乎所有MATLAB函数的分析的基础。在MATLAB7.0中，可以支持多种线性代数中定义的操作，正是其强大的矩阵运算能力才使得MATLAB成为优秀的数值计算软件。本节将主要介绍关于矩阵分析的内容。

 使用norm函数进行范数分析

根据线性代数的知识，对于线性空间中的某个向量$x=\{x_1, x_2, \cdots, x_n\}$，其对应的$P$级范数的定义为$\|x\|_p=(\sum_{i=1}^{n}|x_i|^p)^{1/p}$，其中的参数$p=1, 2, \cdots, n$。同时，为了保证整个定义的完整性，定义范数数值$\|x\|_\infty=\max_{1\le i\le n}|x_i|$，$\|x\|_{-\infty}=\min_{1\le i\le n}|x_i|$。

矩阵范数的定义是基于向量的范数而定义的，具体的表达式为：

$$\|A\|=\max_{\forall x\neq 0}\frac{\|Ax\|}{\|x\|}$$

在实际应用中，比较常用的矩阵范数是1、2和∞阶范数，其对应的定义如下：

$\|A\|_1=\max_{1\le j\le n}\sum_{i=1}^{n}|a_{ij}|$，$\|A\|_2=\sqrt{S_{\max}\{A^TA\}}$和$\|A\|_\infty=\max_{1\le i\le n}\sum_{j=1}^{n}|a_{ij}|$

在上面的定义式$\|A\|_2=\sqrt{S_{\max}\{A^TA\}}$中，$S_{\max}\{A^TA\}$表示矩阵$A$的最大奇异值的平方，关于奇异值的定义将在后面章节中介绍。



之所以在本小节中介绍范数分析的内容，是因为矩阵的范数将直接影响通过MATLAB求解得到的数值解的精度。

在 MATLAB 中，求解向量和矩阵范数的命令如下：

- n = norm(A)　计算向量或者矩阵的 2 阶范数；
- n = norm(A,p)　计算向量或者矩阵的 p 阶范数；



在上面的命令 n = norm(A,p)中，p 可以选择任何大于 1 的实数，如果需要求解的是无穷阶范数，则可以将 p 设置为 inf 或者 -inf。

例 3.1　根据定义和 norm 来分别求解向量的范数。

step 1 进行范数运算。选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
% 输入向量
x=[ 1:6] ;
y=x.^2;
% 使用定义求解各阶范数
N2=sqrt(sum(y));
Ninf=max(abs(x));
Nvinf=min(abs(x));
% 使用 norm 命令求解范数
n2=norm(x);
ninf=norm(x,inf);
nvinf=norm(x,-inf);
% 输出求解的结果
disp('The method of definition:')
fprintf('The 2-norm is %6.4f\n',N2)
fprintf('The inf-norm is %6.4f\n',Ninf)
fprintf('The minusinf-norm is %6.4f\n',Nvinf)
fprintf('\n//--------------------------------//\n\n')
disp('The method of norm command:')
fprintf('The 2-norm is %6.4f\n',n2)
fprintf('The inf-norm is %6.4f\n',ninf)
fprintf('The minusinf-norm is %6.4f\n',nvinf)
```

在输入上面的代码后，将该程序代码保存为 "normex.m" 文件。

step 2 查看运算结果。在 MATLAB 的命令窗口中输入 "normex" 后，按 "Enter" 键，得到计算结果如下：

```
The method of definition:
The 2-norm is 9.5394
The inf-norm is 6.0000
The minusinf-norm is 1.0000

//--------------------------------//

The method of norm command:
The 2-norm is 9.5394
```

```
The inf-norm is 6.0000
The minusinf-norm is 1.0000
```



从上面的结果可以看出，根据范数定义得到的结果和norm命令得到的结果完全相同。通过上面的代码，读者可以更好地理解范数的定义。

例 3.2 根据定义和norm来分别求解Hilbert矩阵的范数。

step 1 进行范数运算。选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开M文件编辑器，在其中输入下面的程序代码：

```
%输入矩阵
A=hilb(5);
%使用定义求解各阶范数
N1=max(sum(abs(A)));
N2=norm(A);
Ninf=max(sum(abs(A')));
Nfro=sqrt(sum(diag(A'*A)));
%使用norm命令求解范数
n1=norm(A,1);
n2=norm(A,2);
ninf=norm(A,inf);
nfro=norm(A,'fro');
%输出求解的结果
disp('The method of definition:')
fprintf('The 1-norm is %6.4f\n',N1)
fprintf('The 2-norm is %6.4f\n',N2)
fprintf('The inf-norm is %6.4f\n',Ninf)
fprintf('The Ff-norm is %6.4f\n',Nfro)
fprintf('\n//-------------------------------//\n\n')
disp('The method of norm command:')
fprintf('The 1-norm is %6.4f\n',n1)
fprintf('The 2-norm is %6.4f\n',n2)
fprintf('The inf-norm is %6.4f\n',ninf)
fprintf('The F-norm is %6.4f\n',nfro)
```

在输入上面的代码后，将该程序代码保存为 "normex2.m" 文件。

step 2 查看运算结果。在MATLAB的命令窗口中输入 "normex2"，然后按 "Enter" 键，得到计算结果如下：

```
The method of definition:
The 1-norm is 2.2833
The 2-norm is 1.5671
The inf-norm is 2.2833
The Ff-norm is 1.5809

//-------------------------------//

The method of norm command:
The 1-norm is 2.2833
The 2-norm is 1.5671
The inf-norm is 2.2833
The F-norm is 1.5809
```

step 3 查看矩阵 *A* 的数值元素。在上面的步骤中，分别使用定义和 norm 命令来求解 Hilbert 矩阵 *A* 的范数，用户可以查看 *A* 的具体元素：

```
%定义数值显示格式
>> format rat
>> A
A =
       1          1/2        1/3        1/4        1/5
      1/2         1/3        1/4        1/5        1/6
      1/3         1/4        1/5        1/6        1/7
      1/4         1/5        1/6        1/7        1/8
      1/5         1/6        1/7        1/8        1/9
```



在 MATLAB 中，Hilbert 矩阵是著名的病态矩阵，主要用来分析矩阵的性能。可以使用 hilb 命令来创建该矩阵，其元素满足等式 $A(i,j)=1/(i+j-1)$。

3.1.2 使用 normest 函数进行范数分析

当需要分析的矩阵比较大时，求解矩阵范数的时间就会比较长，因此当允许某个近似的范数满足某种条件时，可以使用 normest 函数来求解范数。在 MATLAB 的设计中，normest 函数主要是用来处理稀疏矩阵的，但是该命令也可以接受正常矩阵的输入，一般用来处理维数比较大的矩阵。

normest 函数的主要调用格式如下：

- nrm = normest(S)　估计矩阵 S 的 2 阶范数数值，默认的允许误差数值维为 1e-6；
- nrm = normest(S,tol)　使用参数 tol 作为允许的相对误差。

例 3.3　分别使用 norm 和 normest 命令来求解矩阵的范数。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> W = gallery('wilkinson',500) ;
t1=clock;
W_norm=norm(W);
t2=clock;
t_norm=etime(t2,t1);
t3=clock;
W_normest=normest(W);
t4=clock;
t_normest=etime(t4,t3);
```



在上面的程序代码中，首先创建Wilkinson的高维矩阵，然后分别使用norm和normest命令求解矩阵的范数，并统计每个命令所使用的时间。

step 2 查看计算得到的矩阵范数。在命令窗口中输入对应的变量名称，得到的结果如下：

```
W_norm =
  250.2462
t_norm =
    0.7410
```

```
W_normest =
  250.2368
t_normest =
0.3210
```



从上面的结果中可以看出，两种方法得到的结果几乎相等，但在消耗的时间上，normest 命令明显要少于 norm 命令。

step 3 修改矩阵的维度，重新求解范数。在 MATLAB 的命令窗口中输入下面的命令：

```
% 重新设置矩阵的维度
W = gallery('wilkinson',1000) ;
t1=clock;
W_norm=norm(W)
t2=clock;
t_norm=etime(t2,t1)
t3=clock;
W_normest=normest(W)
t4=clock;
t_normest=etime(t4,t3)
% 显示结果
W_norm =
  500.2462
t_norm =
     8.4620
W_normest =
  500.2116
t_normest =
     4.4270
```

从上面的结果中可以看出，维度越大则两个命令求解所消耗时间的差别越大，建议在求解大型矩阵的范数时，使用 normest 命令。



关于每个命令的计算时间，和运行命令的硬件环境以及是否首次运行软件有关，因此读者运行的时间结果可能和上面的结果不同，但是两者之间的大小关系不会变化。

3.1.3 条件数分析

在线性代数中，描述线性方程 $Ax=b$ 的解对 b 中的误差或不确定性的敏感度的度量就是矩阵 A 的条件数，其对应的数学定义是：

$$k=\|A^{-1}\|\cdot\|A\|$$

根据基础的数学知识，矩阵的条件数总是大于等于 1。其中，正交矩阵的条件数为 1，奇异矩阵的条件数为 ∞，而病态矩阵的条件数则比较大。

依据条件数，方程解的相对误差可以由下面的不等式来估计：

$$\frac{1}{k}\left(\frac{|\delta b|}{b}\right)\leqslant\frac{|\delta x|}{|x|}\leqslant k\left(\frac{|\delta b|}{b}\right)$$

在 MATLAB 中，求取矩阵 X 的条件数的命令如下：

c = cond(X) 求矩阵 X 的条件数

例3.4 以MATLAB产生的Magic和Hilbert矩阵为例，使用矩阵的条件数来分析对应的线性方程解的精度。

step 1 进行数值求解。在MATLAB的命令窗口中输入下面的命令：

```
>> M=magic(5);
>> b=ones(5,1);
%利用左除M求解近似解
>> x=M\b;
%利用准确的求解
>> xinv=inv(M)*b;
%计算实际相对误差
>> ndb=norm(M*x-b);
>> nb=norm(b);
>> ndx=norm(x-xinv);
>> nx=norm(x);
>> er=ndx/nx;
>> k=cond(M);
%计算最大可能的近似相对误差
>> erk1=k*eps;
%计算最大可能的相对误差
>> erk2=k*ndb/nb;
```



在上面的程序代码中，首先产生Magic矩阵，然后使用近似解和准确解进行比较，得出计算误差。

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
>> k
k =
     5.4618
>> er
er =
   2.9403e-016
>> erk1
erk1 =
   1.2128e-015
>> erk2
erk2 =
   6.6426e-016
```



从上面的结果可以看出，矩阵M的条件数5.4618，这种情况下引起的计算误差是很小的，其误差是完全可以接受的。

step 3 修改求解矩阵，重新计算求解的精度。在命令窗口中输入下面的代码：

```
M=hilb(12);
b=ones(12,1);
x=M\b;
xinv=invhilb(12)*b;
ndb=norm(M*x-b);
```

```
nb=norm(b);
ndx=norm(x-xinv);
nx=norm(x);
er=ndx/nx;
k=cond(M);
erk1=k*eps;
erk2=k*ndb/nb;
```

step 4 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
>> k
k =
  1.7945e+016
>> er
er =
    0.0119
>> erk1
erk1 =
    3.9846
>> erk2
erk2 =
  5.3479e+007
```



从上面的结果可以看出，该矩阵的条件数为1.7945e+016，该矩阵在数学理论中就是高度病态的，这样会造成比较大的计算误差。

3.1.4 数值矩阵的行列式

在 MATLAB 中，求解矩阵行列式的命令比较简单，其调用格式如下：

- d = det(X)　　求解矩阵 X 的行列式，如果输入的参数 X 不是矩阵，而是一个常数，则该命令返回原来的常数。

例 3.5　求解矩阵的行列式。

step 1 求解矩阵的行列式。在 MATLAB 的命令窗口中输入下面的命令：

```
for i=1:3
A=randint(i+2);
a(i)=det(A);
disp('The Matrix is:');
disp(A);
disp('The determinant is');
disp(num2str(a(i)));
end
```

step 2 查看求解的结果。在输入上面的程序代码后，按"Enter"键，得到的结果如下：

```
The Matrix is:
     1     0     0
     0     0     1
     0     0     0
```

```
The determinant is
0
The Matrix is:
     0     1     0     0
     1     1     0     1
     0     0     1     0
     1     0     1     0
The determinant is
1
The Matrix is:
     0     0     1     1     1
     1     1     0     0     1
     0     1     1     1     0
     1     0     1     1     1
     0     1     1     0     1
The determinant is
-2
```



在上面的程序代码中，首先使用randint命令产生随机矩阵，然后使用det命令来计算这些矩阵的行列式。

3.1.5 符号矩阵的行列式

需要说明的是，det命令除了可以计算数值矩阵的行列式之外，还可以计算符号矩阵的行列式，下面举例说明。

例3.6 求解符号矩阵的行列式。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> syms t;
>>A=[ sin(t),cos(t);-cos(t),sin(t)];
>> B=det(A);
>> C=simple(B);
```



在上面的程序代码中，使用det命令求解矩阵A的行列式表达式B，然后使用simple命令来化简表达式B。

step 2 查看运行的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
A =
 [   sin(t),  cos(t)]
[  -cos(t),  sin(t)]
B =
 sin(t)^2+cos(t)^2
C =
 1
```



从上面的结果可以看出，使用det命令也可以求解符号矩阵的行列式。关于符号运算的其他命令，读者可以查看符号运算的章节。



矩阵的化零矩阵

对于非满秩的矩阵 A，存在某矩阵 Z，满足 $A \cdot Z=0$，同时矩阵 Z 是一个正交矩阵，也就是说 $Z' \cdot Z=I$，则矩阵 Z 被称为矩阵 A 的化零矩阵。在 MATLAB 中，求解化零矩阵的命令为 null，其具体的调用格式如下：

- Z = null(A)　返回矩阵 A 的化零矩阵，如果化零矩阵不存在则返回空矩阵；
- Z = null(A,'r')　返回有理数形式的化零矩阵。

例 3.7　求解非满秩矩阵 A 的化零矩阵。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> A = [ 1      2      3
     4      5      6
     7      8      9] ;
>> Z = null(A);
R=A* Z;
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
>> Z
Z =
   -0.4082
    0.8165
   -0.4082
>> R
R =
  1.0e-015 *
         0
   -0.4441
         0
```

step 2 求解有理数形式的化零矩阵。在 MATLAB 的命令窗口中输入下面的命令：

```
>> ZR=null(A,'r');
>> RZ=A* ZR;
```

step 1 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
>> ZR
ZR =
     1
    -2
     1
>> RZ
RZ =
     0
     0
     0
```

线性方程组

线性方程组是线性代数中的主要内容之一，也是理论发展最为完整的部分。在 MATLAB 中也包含

多种处理线性方程组的命令，限于篇幅，本节不详细展开各种理论和对应的命令，主要处理三种方程：恰定方程组、超定方程组和欠定方程组，下面详细介绍。

3.2.1 非奇异线性方程组

在线性代数中，恰定方程组是指方程组的个数和未知量个数相等的方程组，在恰定方程组中$\boldsymbol{A}$是方阵，矩阵$\boldsymbol{B}$可能是向量也可能是方阵。对于恰定方程组，MATLAB 提供一个十分方便的命令，左除"\"。根据系数矩阵$\boldsymbol{A}$的奇异属性，该命令可以得到不同的结果：

- 如果恰定方程是非奇异的，则左除命令给出恰定方程组的精确解；
- 如果恰定方程组是奇异的，MATLAB 会显示警告信息，同时给出解 NaN。

下面分别用例子来详细说明如何使用该命令求解方程组。

例 3.8　求解非奇异矩阵的线性方程的解。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> A = pascal(4);
>> b = [1; 3; 4;6];
>> x=A\b;
>> Bsol=A*x;
>> D=det(A);
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
A =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
D =
     1
x =
    -4
    10
    -7
     2
Bsol =
     1
     3
     4
     6
```

从上面的结果可以看出，方程组的系数矩阵$\boldsymbol{A}$的行列式为 1，则系数矩阵$\boldsymbol{A}$是非奇异的，因此 MATLAB 可以给出准确的数值解。同时，该数值解和系数矩阵相乘，得到的结果和原来的数值完全相同。



在上面的代码中，首先使用 pascal 命令创建数值矩阵，该命令是 MATLAB 的内置函数，该命令将产生一个对称的正定矩阵，矩阵的数值取自 Pascal 三角形。

3.2.2 奇异线性方程组

例 3.9 求解奇异矩阵的线性方程的解。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> A =[ 1     3     7
-1     4     4
1    10    18 ];
>> b=[ 6;4;15] ;
>> x=A\b
Warning: Matrix is singular to working precision.
x =
   NaN
   Inf
  -Inf
```



从上面的结果可以看出，MATLAB会显示提示信息，表示该矩阵是奇异矩阵，因此无法得到精确的数值解。

step 2 查看矩阵信息。可以使用多种命令来查看矩阵 A 是否是奇异的：

```
>> det(A)
ans =
     0
>> rank(A)
ans =
     2
```



上面的结果可以看出，矩阵 A 的行列式为0，同时矩阵的秩为2，表示该矩阵 ***A*** 是严格奇异的，MATLAB 将无法给出精确的数值解。

对于恰定方程组中，如果数值解有解，则可以使用矩阵 ***A*** 的伪逆矩阵 pinv(A)来得到方程的一个解，其对应的数值解为 pinv(A)*B，下面举例来说明。

例 3.10 使用伪逆矩阵的方法求解奇异矩阵的线性方程的解。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> A =[ 1     3     7
-1     4     4
1    10    18 ];
>> b=[ 5;2;12] ;
>> x=pinv(A)*b;
>> bsol=A*x;
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
x =
    0.3850
   -0.1103
    0.7066
bsol =
```

```
  5.0000
  2.0000
 12.0000
```



从上面的结果可以看出，通过使用伪逆矩阵的方法，可以求解得到数值解，同时该数值解可以精确地满足结果。

step 3 修改需要求解的数值。在MATLAB的命令窗口中输入下面的命令：

```
>> A = [ 1      3      7
-1      4      4
1     10     18 ];
>> b=[ 6;4;15];
>> x=pinv(A)*b;
>> bsol=A*x;
```

step 4 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
x =
   0.0759
    0.3126
    0.6647
bsol =
    5.6667
    3.8333
   15.1667
```



从上面的结果可以看出，通过上面的方法求解得到的结果并不完全满足原来的数值条件，并不具有上面步骤中的精度。

3.2.3 欠定线性方程组

在线性代数的理论中，欠定线性方程组是指方程组的未知量个数多于方程个数的问题，这类问题的解不是唯一解，MATLAB7.0将首先寻求一个基本解，然后再寻求非零解。从解法的角度来看，MATLAB7.0采用的是QR分解的方法来求解欠定线性方程组。下面举例详细说明。

例3.11 求解欠定线性方程组。

step 1 在MATLAB的命令窗口中输入下面的命令：

```
>> A = [ 1      2      3
         4      5      6
         7      8      9
        10     11     12 ];
>>b = [ 1;3;5;7];
>> [ Q,R]  = qr(A);
>>y=Q'*b;
>>xqr = R\y;
>> x=A\b;
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
Warning: Rank deficient, rank = 2,  tol =   1.4594e-014.
x =
    0.5000
         0
    0.1667
Warning: Rank deficient, rank = 2,  tol =   1.4594e-014.
xqr =
    0.5000
         0
    0.1667
Q =
   -0.0776   -0.8331    0.5456   -0.0478
   -0.3105   -0.4512   -0.6919    0.4704
   -0.5433   -0.0694   -0.2531   -0.7975
   -0.7762    0.3124    0.3994    0.3748
R =
  -12.8841  -14.5916  -16.2992
         0   -1.0413   -2.0826
         0         0   -0.0000
         0         0         0
```



从上面的结果可以看出，直接通过左除求解得到的数值解和使用QR分析得到的数值解是完全相同的，因此读者可以了解到MATLAB求解欠定方程组的原理。

3.2.4 超定线性方程组

超定线性方程组是指方程组的个数比未知数个数多的情况，对于这种情况。MATLAB提供伪逆矩阵的方法来求解，关于该方法在前面已经介绍过，在本小节中将利用一个具体的实例来说明如何求解超定线性方程组。

例 3.12 求解超定线性方程组。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> A = magic(8);
>>A = A(:,1:6) ;
>> b = 260*ones(8,1);
>> x = pinv(A)*b ;
>>bsol=A*x;
>> xs=A\b;
>>bs=A*xs;
>> nx=norm(x);
>> nxs=norm(xs);
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
A =
    64     2     3    61    60     6
     9    55    54    12    13    51
    17    47    46    20    21    43
    40    26    27    37    36    30
    32    34    35    29    28    38
    41    23    22    44    45    19
```

```
    49    15    14    52    53    11
     8    58    59     5     4    62
x =
    1.1538
    1.4615
    1.3846
    1.3846
    1.4615
1.1538

Warning: Rank deficient, rank = 3,  tol =   1.8829e-013.
xs =
    4.0000
    5.0000
         0
         0
         0
   -1.0000
bsol =
  260.0000
  260.0000
  260.0000
  260.0000
  260.0000
  260.0000
  260.0000
  260.0000
bs =
  260.0000
  260.0000
  260.0000
  260.0000
  260.0000
  260.0000
  260.0000
  260.0000
nx =
    3.2817
nxs =
    6.4807
```



从上面的结果可以看出，尽管左除和伪逆矩阵的方法求解得到的数值解都具有足够的精度，但是使用伪逆矩阵得到的数值解的范数要小。

3.3 矩阵分解

矩阵分解主要是指将一个矩阵分解为几个比较简单的矩阵连乘的形式。无论在理论上还是在工程应用上，矩阵分解都是十分重要的。在MATLAB中，线性方程组的求解主要基于三种基本的矩阵分解，Cholesky分解、LU分解和QR分解。对于这些分解，MATLAB都提供对应的函数。除了上面介绍的几种分解之外，本节还将介绍奇异值分解和舒尔求解两种比较常见的分解。

3.3.1 Cholesky 分解

Cholesky 分解是把一个对称正定矩阵 ***A*** 分解为一个上三角矩阵 ***R*** 和其转置矩阵的乘积，其对应的表达式为：***A=R' R***。从理论的角度来看，并不是所有的对称矩阵都可以进行 Cholesky 分解的，需要进行 Cholesky 分解的矩阵必须是正定的。

在 MATLAB 中，进行 Cholesky 分解的是 chol 命令：

- R=chol(X) 其中 ***X*** 是对称的正定矩阵，***RR*** 是上三角矩阵，使得 ***X=R'R***。如果矩阵 ***X*** 是非正定矩阵，该命令会返回错误信息；
- [R,p] = chol(X) 该命令返回两个参数，并不返回错误信息。当 ***X*** 是正定矩阵时，返回的矩阵 ***R*** 是上三角矩阵，而且满足等式 ***X=R'R***，同时返回参数 p=0；当 ***X*** 不是正定矩阵时，返回的参数 p 是正整数，***R*** 是三角矩阵，且矩阵阶数是 p－1，并且满足等式 ***X***(1:*p*-1, 1:*p*-1)=***R'R***。



对对阵正定矩阵进行分解在矩阵理论中是十分重要的理论，可以首先对该对称正定进行Cholesky分解，然后经过处理得到线性方程的解，这些内容将在后面的步骤中通过实例来介绍。

例 3.13 对对称正定矩阵进行 Cholesky 分解。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> n = 5;
>>X = pascal(n)
>>R = chol(X)
>>C=transpose(R)*R
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
X =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
R =
     1     1     1     1     1
     0     1     2     3     4
     0     0     1     3     6
     0     0     0     1     4
     0     0     0     0     1
C =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
```



从上面的结果中可以看出，***R*** 是上三角矩阵，同时满足等式 ***C=R'R=X***，表明上面的 Cholesky 分解过程成功。

step 3 修改矩阵信息。在 MATLAB 的命令窗口中输入下面的命令：

```
>> X(n,n) = X(n,n)-1
>>[R1,p]=chol(X)
>>C1=transpose(R1)*R1
>>C2=X(1:p-1,1:p-1)
```

step 4 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
X =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    69
R1 =
     1     1     1     1
     0     1     2     3
     0     0     1     3
     0     0     0     1
p =
     5
C1 =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
C2 =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
```



从上面的结果中可以看出，当用户将原来正定矩阵的最后一个元素减 1 后，矩阵将不是正定矩阵，并且满足条件 $X(1:p-1, 1:p-1)=R'R$。

3.3.2 使用 Cholesky 分解求解方程组

例 3.14 使用 Cholesky 分解来求解线性方程组。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> A=pascal(4);
>>b=[1;4;6;13];
>>x=A\b
>>R=chol(A);
>>Rt=transpose(R);
>>xr=R\(Rt\b)R
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
x =
```

```
     -9
     23
    -19
      6
xr =
     -9
     23
    -19
      6
```

从上面的结果可以看出，使用Cholesky分解求解得到的线性方程组的数值解，与使用左除得到的结果完全相同。其对应的数学原理如下：

对于线性方程组$\boldsymbol{A}x=\boldsymbol{b}$，其中$\boldsymbol{A}$是对称的正定矩阵，其$\boldsymbol{A}=\boldsymbol{R}'\boldsymbol{R}$，则根据上面的定义，线性方程组可以转换为$\boldsymbol{R}'\boldsymbol{R}x=b$，该方程组的数值为$x=\boldsymbol{R}\backslash(\boldsymbol{R}'\backslash b)$。



尽管两个方法得到的结果相同，但是其复杂度却不相同。假设$\boldsymbol{A}$是$n\times n$的方阵，则命令chol的计算复杂度是$o(n^3)$，而左除命令“\”的计算复杂度为$o(n^2)$。

3.3.3 不完全Cholesky分解

对于稀疏矩阵，MATLAB提供cholinc命令来做不完全的Cholesky分解，该命令的另外一个重要功能是求解实数半正定矩阵的Cholesky分解，其具体的调用格式如下：

- R = cholinc(X,droptol) 其中参数X和R的含义和chol命令中的含义相同，其中droptol表示不完全Cholesky分解的丢失容限，当该参数为0时，则属于完全Cholesky分解。
- R = cholinc(X,options) 其中参数options用来设置该命令的相关参数；具体地讲，options是一个结构体，包含droptol、michol和rdiag三个参数。
- R = cholinc(X,'0') 完全Cholesky分解。
- [R,p] = cholinc(X,'0') 和命令chol(X)相同。
- R = cholinc(X,'inf') 采用Cholesky-Infinity方法来进行分解，Cholesky-Infinity方法是基于Cholesky分解的，但是可以用来处理实半正定分解。

例3.15 使用cholinc命令对矩阵进行Cholesky分解。

step 1 在MATLAB的命令窗口中输入下面的命令：

```
>> H20 = sparse(hilb(20));
>> [R,p] = chol(H20);
>> Rinf = cholinc(H20,'inf');
>> Rfull=full(Rinf(14:end,14:end))
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
Rfull =
   Inf     0     0     0     0     0     0
     0   Inf     0     0     0     0     0
     0     0   Inf     0     0     0     0
     0     0     0   Inf     0     0     0
     0     0     0     0   Inf     0     0
```

```
     0     0     0     0     0   Inf     0
     0     0     0     0     0     0   Inf
```

step 3 检验是否满足分解条件。在 MATLAB 的命令窗口中输入下面的命令：

```
>> H=full(H20(14:end,14:end));
>>H20R=Rfull'*Rfull;
```

step 4 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
H =
    0.0370    0.0357    0.0345    0.0333    0.0323    0.0313    0.0303
    0.0357    0.0345    0.0333    0.0323    0.0313    0.0303    0.0294
    0.0345    0.0333    0.0323    0.0313    0.0303    0.0294    0.0286
    0.0333    0.0323    0.0313    0.0303    0.0294    0.0286    0.0278
    0.0323    0.0313    0.0303    0.0294    0.0286    0.0278    0.0270
    0.0313    0.0303    0.0294    0.0286    0.0278    0.0270    0.0263
    0.0303    0.0294    0.0286    0.0278    0.0270    0.0263    0.0256
H20R =
   Inf   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   Inf   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   Inf   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   Inf   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   Inf   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   Inf   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   Inf
```



从上面的结果可以看出，尽管cholinc命令可以求解得到分解结果，但是该分解结果并不能保证开始的等式关系。

3.3.4 LU 分解

LU分解又被称为是高斯消去法。它可以将任意一个方阵 $\boldsymbol{A}$ 分解为一个“心理”下三角矩阵 $\boldsymbol{L}$ 和一个上三角矩阵 $\boldsymbol{U}$ 的乘积，也就是 $\boldsymbol{A}=\boldsymbol{LU}$。其中，“心理”下三角矩阵的定义为下三角矩阵和置换矩阵的乘积。

在 MATLAB 中，求解 LU 分解的命令为 lu，其主要调用格式如下：

- [L,U] = lu(X) 其中 X 是任意方阵，L 是“心理”下三角矩阵，U 是上三角矩阵，这三个变量满足的条件式为 $\boldsymbol{X}=\boldsymbol{LU}$；
- [L,U,P] = lu(X) 其中 X 是任意方阵，L 是“心理”下三角矩阵，U 是上三角矩阵，P 是置换矩阵，满足的条件式为 $\boldsymbol{PX}=\boldsymbol{LU}$；
- Y = lu(X) 其中 X 是任意方阵，把上三角矩阵和下三角矩阵合并在矩阵 Y 中给出，满足等式为 $\boldsymbol{Y}=\boldsymbol{L}+\boldsymbol{U}-\boldsymbol{I}$，该命令将损失置换矩阵 $\boldsymbol{P}$ 的信息。

例 3.16 使用 lu 命令对矩阵进行 LU 分解。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> A = [ -1    8    -5
9    -1    2
```

```
              2     -5     7 ];
>> [ L1,U1] =lu(A);
>> A1=L1*U1;
>> x=inv(A);
>> x1=inv(U1)*inv(L1);
>> d=det(A);
>> d1=det(L1)*det(U1);
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
L1 =
   -0.1111    1.0000         0
    1.0000         0         0
    0.2222   -0.6056    1.0000
U1 =
    9.0000   -1.0000    2.0000
         0    7.8889   -4.7778
         0         0    3.6620
A1 =
    -1     8    -5
     9    -1     2
     2    -5     7
x =
   -0.0115    0.1192   -0.0423
    0.2269   -0.0115    0.1654
    0.1654   -0.0423    0.2731
x1 =
   -0.0115    0.1192   -0.0423
    0.2269   -0.0115    0.1654
    0.1654   -0.0423    0.2731
d =
  -260
d1 =
  -260
```

从上面的结果可以看出，方阵的 LU 分解满足下面的等式条件：

$$A=LU、U^{-1}L^{-1}=A^{-1} \text{和} \det(A)=\det(L)\det(U)$$

step 3 使用三个输出变量的命令形式。在 MATLAB 的命令窗口中输入下面的命令：

```
>> [ L,U,P]  = lu(A);
>> Lp=P*L;
>> Ap=L*U;
>> Pa=P*A;
```

step 3 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
L =
    1.0000         0         0
   -0.1111    1.0000         0
    0.2222   -0.6056    1.0000
U =
    9.0000   -1.0000    2.0000
         0    7.8889   -4.7778
         0         0    3.6620
```

```
P =
     0     1     0
     1     0     0
     0     0     1
Ap =
     9    -1     2
    -1     8    -5
     2    -5     7
Pa =
     9    -1     2
    -1     8    -5
     2    -5     7
Lp =
   -0.1111    1.0000         0
    1.0000         0         0
    0.2222   -0.6056    1.0000
```

从上面的结果可以看出，使用三个输出变量的命令满足下面的等式关系：

$$PA=LU \text{ 和 } PL=L'$$

在上面的等式中，L'表示使用两个输出变量求解的LU分解矩阵结果。

step 5 使用LU分解来求解线性方程组。在MATLAB的命令窗口中输入下面的命令：

```
\>> b=[2;3;5];
>>xb=A\b;
>>y1 = L\b;
>>xb1=U\y1;
>> y2 = L1\b;
>>xb2=U1\y1;
```

step 6 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
xb =
    0.1231
    1.2462
    1.5692
xb1 =
    0.1231
    1.2462
    1.5692
xb2 =
    0.1231
    1.2462
    1.5692
```



从上面的结果可以看出，无论两个输出变量还是三个输出变量的命令得到的结果，都和使用左除命令得到的结果相同。

3.3.5 不完全LU分解

对于稀疏矩阵，MATLAB提供函数luinc来进行不完全的LU分解。其调用格式如下：

- [L,U] = luinc(X,droptol) 命令中各参数 X 和 R 的含义和 lu 命令中的含义相同，其中 droptol 表示不完全 LU 分解的丢失容限，当该参数为 0 时，则属于完全 LU 分解；
- [L,U] = luinc(X,options) 参数 options 设置关于 LU 分解的各种参数；
- [L,U] = luinc(X,'0') 0 级不完全 LU 分解；
- [L,U,P] = luinc(X,'0') 0 级不完全 LU 分解。

例 3.17 使用 lunic 命令对稀疏矩阵进行 LU 分解。

step 1 加载稀疏矩阵，并绘制稀疏矩阵图形。在 MATLAB 的命令窗口中输入下面的命令：

```
%加载稀疏矩阵
>> load west0479;
>> S = west0479;
>> LU = lu(S);
%绘制稀疏矩阵的图形
>> subplot(1,2,1);
>> spy(S);
>>title('S')
%使用 LU 求解得到的结果
>> subplot(1,2,2);
>> spy(LU);
>>title('LU')
```

step 2 查看求解的结果。在输入上面的程序代码后，按 "Enter" 键，得到的图形如图 3.1 所示。

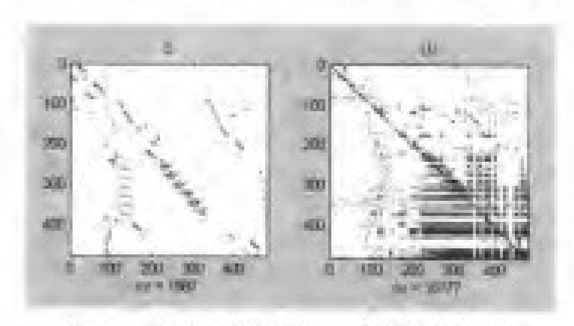


图 3.1 系数矩阵和 LU 分解结果图形



由于加载的系数矩阵维度比较大，如果直接使用数据查看，很难看出或者分辨出矩阵的性质。在上面的实例中，使用了 spy 命令来查看矩阵的属性。

step 3 使用 lunic 命令对稀疏矩阵进行 LU 分解。在命令窗口中输入下面的命令：

```
%进行 0 级 LU 分解
>> [ L,U,P] = luinc(S,'0');
>> D = (L*U).*spones(P*S)-P*S;
%打开新的图形窗口
%绘制使用 lunic 命令得到的结果
>>figure
>>subplot(221);
>>spy(L);
>>title('L:luinc(S,0)')
>> subplot(222);
>>spy(U);
```

```
>>title('U:luinc(S,0)')
>>subplot(223);
>>spy(P*S);
>>title('P*S')
>>subplot(224);
>>spy(L*U);
>>title('L*U')
```

step 4 查看求解的结果。在输入上面的程序代码后，按"Enter"键，得到的图形如图 3.2 所示。

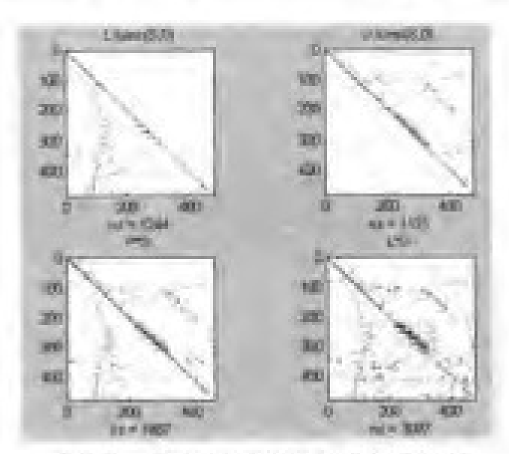


图 3.2　使用 lunic 命令得到的结果

step 5 使用不同的误差容忍度对稀疏矩阵进行 LU 分解。在命令窗口中输入下面的命令：

```
>>[ IL1,IU1,IP1] = luinc(S,1e-8);
>> [ IL2,IU2,IP2] = luinc(S,1e-4);
Warning: Incomplete upper triangular factor has 7 zero diagonals.
         It cannot be used as a preconditioner for an iterative method
>> [ IL3,IU3,IP3] = luinc(S,1e-2);
Warning: Incomplete upper triangular factor has 22 zero diagonals.
         It cannot be used as a preconditioner for an iterative method
>> [ IL4,IU4,IP4] = luinc(S,1);
Warning: Incomplete upper triangular factor has 87 zero diagonals.
         It cannot be used as a preconditioner for an iterative method
>> figure;
>> subplot(221);
>> spy(IL1*IU1);
>> title('luinc(S,1e-8)')
>> subplot(222);
spy(IL2*IU2);
title('luinc(S,1e-4)')
>> subplot(223);
spy(IL3*IU3);
title('luinc(S,1e-2)')
>> subplot(224);
spy(IL4*IU4);
title('luinc(S,1)')
```

step 6 查看求解的结果。在输入上面的程序代码后，按"Enter"键，得到的图形如图 3.3 所示。

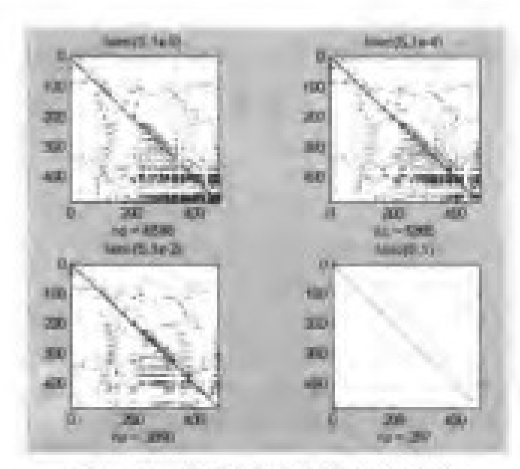


图 3.3 使用不同的误差容忍度

step 7 绘制“droptol-nnz”图形。在本命令中，droptol 表示 LU 分解的丢失容限，nnz 则表示系数矩阵中非零元素的个数，用户可以绘制两个变量之间的关系。在 MATLAB 的命令窗口中输入下面的代码：

```
>> nz(1)=nnz(IL1);
tol=1.e-8;
nz(1)=nnz(IL1);
tol(1)=1.e-8;
nz(2)=nnz(IL2);
tol(2)=1.e-4;
nz(3)=nnz(IL3);
tol(3)=1.e-2;
nz(4)=nnz(IL4);
tol(4)=1;
>> semilogx(tol,nz,'g','LineWidth',1.5)
>> set(gca,'Ytick',[0:650:7800])
set(gca,'Ylim',[0 7800])
>> title('Drop tolerance vs nnz(luinc(S,droptol))')
xlabel('Drop tolerance')
ylabel('nnz')
>> grid
```

step 8 查看求解的结果。在输入上面的程序代码之后，按“Enter”键，得到的结果如图 3.4 所示。

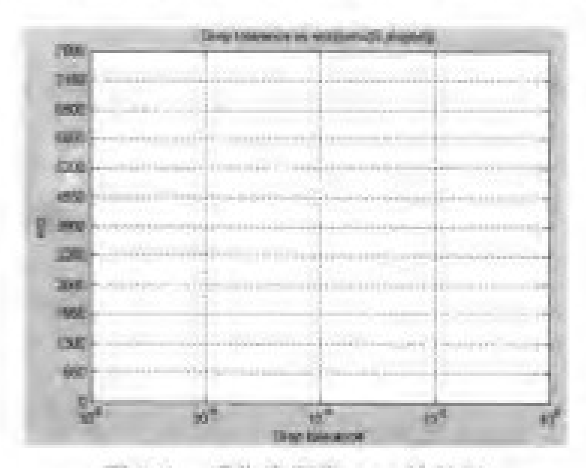


图 3.4 丢失容限和 nnz 的关系

step 9 绘制"droptol-norm"图形。在本命令中，droptol表示LU分解的丢失容限，norm则表示LU分解的相对误差，用户可以绘制两个变量之间的关系。在MATLAB的命令窗口中输入下面的代码：

```
%计算相对误差
>> normlu(1)=norm(IL1*IU2-IP1*S,1)/norm(S,1);
>> normlu(1)=norm(IL1*IU1-IP1*S,1)/norm(S,1);
>> normlu(2)=norm(IL2*IU2-IP2*S,1)/norm(S,1);
>> normlu(3)=norm(IL3*IU3-IP3*S,1)/norm(S,1);
>> normlu(4)=norm(IL4*IU4-IP4*S,1)/norm(S,1);
>> loglog(tol,normlu,'g','LineWidth',1.5)
>> title('Drop tolerance vs norm norm(L*U-P*S,1)/norm(S,1))')
>>xlabel('Drop tolerance')
>>ylabel('norm')
>> grid
```

step 10 查看求解的结果。在输入上面的代码后，按"Enter"键，得到的图形如图3.5所示。

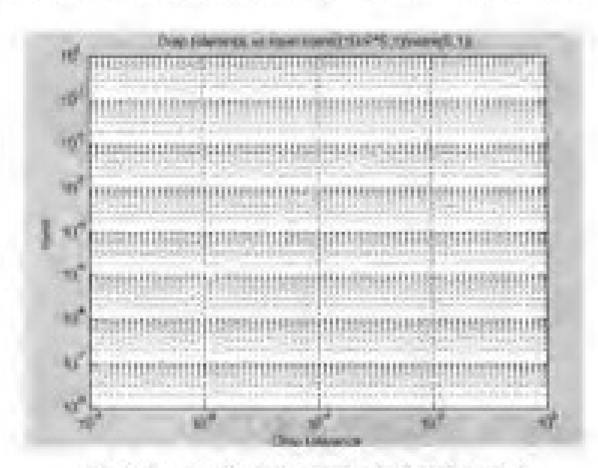


图3.5 丢失容限和相对误差的图形

3.3.6 QR分解

矩阵的正交分解又被称为QR分解，也就是将一个$m \times n$的矩阵$\boldsymbol{A}$分解为一个正交矩阵$\boldsymbol{Q}$和一个上三角矩阵$\boldsymbol{R}$的乘积，也就是说$\boldsymbol{A}=\boldsymbol{QR}$。

在MATLAB中，进行QR分解的命令为qr，其调用格式如下：

- [Q,R] = qr(A) 矩阵$\boldsymbol{R}$和矩阵$\boldsymbol{A}$的大小相同，$\boldsymbol{Q}$是正交矩阵，满足等式$\boldsymbol{A}=\boldsymbol{QR}$，该调用方式适用于满矩阵和稀疏矩阵；
- [Q,R] = qr(A,0) 比较经济类型的QR分解。假设矩阵$\boldsymbol{A}$是一个$m \times n$的矩阵，其中$m>n$，则命令将只计算前n列的元素，返回的矩阵$\boldsymbol{R}$是$n \times n$矩阵；如果$m \leq n$，该命令和上面的命令[Q,R] = qr(A)相等，该调用方式适用于满矩阵和稀疏矩阵；
- [Q,R,E] = qr(A) 该命令中，$\boldsymbol{Q}$是正交矩阵，$\boldsymbol{R}$是上三角矩阵，$\boldsymbol{E}$是置换矩阵，满足条件关系式$\boldsymbol{A} \cdot \boldsymbol{E}=\boldsymbol{Q} \cdot \boldsymbol{R}$，该调用方式适用于满矩阵。

例3.18 使用qr命令对矩阵进行QR分解。

step 1 在MATLAB的命令窗口中输入下面的命令：

```
>> A = magic(5);
```

```
[Q,R] = qr(A)
C=Q*R
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
Q =
   -0.5234    0.5058    0.6735   -0.1215   -0.0441
   -0.7081   -0.6966   -0.0177    0.0815   -0.0800
   -0.1231    0.1367   -0.3558   -0.6307   -0.6646
   -0.3079    0.1911   -0.4122   -0.4247    0.7200
   -0.3387    0.4514   -0.4996    0.6328   -0.1774
R =
  -32.4808  -26.6311  -21.3973  -23.7063  -25.8615
         0   19.8943   12.3234    1.9439    4.0856
         0         0  -24.3985  -11.6316   -3.7415
         0         0         0  -20.0982   -9.9739
         0         0         0         0  -16.0005
C =
   17.0000   24.0000    1.0000    8.0000   15.0000
   23.0000    5.0000    7.0000   14.0000   16.0000
    4.0000    6.0000   13.0000   20.0000   22.0000
   10.0000   12.0000   19.0000   21.0000    3.0000
   11.0000   18.0000   25.0000    2.0000    9.0000
```

从上面的结果可以看出，矩阵***R***是上三角矩阵，同时满足等式***A=QR***，在下面的步骤中，将需要证明***Q***矩阵是正交矩阵。

step 3 证明矩阵***Q***的正交性。在MATLAB的命令窗口中输入下面的命令：

```
>> detQ=det(Q);
>> for i=1:4
A=Q(:,i);
for j=(i+1):5
B=Q(:,j);
C=A'*B;
disp(num2str(C))
end
end
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
detQ =
    1.0000
C=
5.5511e-017
5.5511e-017
-2.7756e-017
6.9389e-018
```

```
2.7756e-017
0
1.3878e-017
0
-6.9389e-017
-2.2204e-016
```

从上面的结果可以看出，矩阵Q的行列式是1，同时，矩阵中每列数据之间的点乘结果都近似为0，因此Q是正交矩阵。



由于Q是正交矩阵，因此经过QR分解得到的矩阵Q和原来的矩阵A是等秩的，读者可以通过分析R的秩来计算矩阵A的秩。

3.3.7 操作QR分解结果

在MATLAB中，除了提供qr命令之外，还提供qrdelete和qrinsert命令来处理矩阵运算的QR分解。其中，qrdelete的功能是删除QR分解得到矩阵的行或者列；qrinsert的功能则是插入QR分解得到矩阵的行或者列。下面以qrdelete命令为例，来说明如何调用命令：

- [Q1,R1] = qrdelete(Q,R,j) 返回矩阵A_1的QR分解结果，其中A_1结果是矩阵A删除第j列得到的结果，而矩阵$A = QR$；
- [Q1,R1] = qrdelete(Q,R,j,'col') 计算结果和[Q1,R1] = qrdelete(Q,R,j)相同；
- [Q1,R1] = qrdelete(Q,R,j,'row') 返回矩阵A_1的QR分解结果，其中A_1结果是矩阵A删除第j行的数据得到的结果，而矩阵$A = QR$。

例3.19 对矩阵QR分解得到的矩阵进行删除运算。

step 1 在MATLAB的命令窗口中输入下面的命令：

```
A = magic(5);
[Q,R] = qr(A);j = 3;
[Q1,R1] = qrdelete(Q,R,j,'row');
C=Q1*R1;
A2 = A;
A2(j,:) = [];
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
Q1 =
    0.5274   -0.5197   -0.6697   -0.0578
    0.7135    0.6911    0.0158    0.1142
    0.3102   -0.1982    0.4675   -0.8037
    0.3413   -0.4616    0.5768    0.5811
R1 =
   32.2335   26.0908   19.9482   21.4063   23.3297
         0  -19.7045  -10.9891    0.4318   -1.4873
         0         0   22.7444    5.8357   -3.1977
         0         0         0  -14.5784    3.7796
C =
   17.0000   24.0000    1.0000    8.0000   15.0000
```

```
   23.0000    5.0000    7.0000   14.0000   16.0000
   10.0000   12.0000   19.0000   21.0000    3.0000
   11.0000   18.0000   25.0000    2.0000    9.0000
A2 =
    17    24     1     8    15
    23     5     7    14    16
    10    12    19    21     3
    11    18    25     2     9
```

在上面的结果中，满足等式，其中$C=Q_1 \cdot R_1$就是矩阵A删除对应数据行的结果，也就是上面结果中的A_2矩阵。

step 3 证明Q_1矩阵的正交性。在MATLAB的命令窗口中输入下面的命令：

```
>> detQ1=det(Q1);
>> for i=1:3
A=Q1(:,i);
for j=(i+1):4
B=Q1(:,j);
C=A'*B;
disp(num2str(C))
end
end
```

step 4 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
detQ1 =
    1.0000
C=
2.7756e-017
1.1102e-016
0
-5.5511e-017
0
0
```

从上面的结果可以看出，矩阵Q_1的行列式是1，同时，矩阵中每列数据之间的点乘结果都近似为0，因此Q_1是正交矩阵。



在MATLAB中，qrinsert命令和qrdelete命令的调用方法几乎相同，在这里就不详细介绍该命令的用法了。下面将用一个简单的例子说明如何使用qrinsert命令。

例3.20 对矩阵QR分解得到的矩阵进行插入运算。

step 1 在MATLAB的命令窗口中输入下面的命令：

```
>> A = magic(5);
[Q,R] = qr(A);
j = 3;
x = 1:5;
[Q1,R1] = qrinsert(Q,R,j,x,'row');
>> Aqr=Q1*R1;
>> A2 = [A(1:j-1,:); x; A(j:end,:)];
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
Q1 =
    0.5231    0.5039   -0.6750    0.1205    0.0411    0.0225
    0.7078   -0.6966    0.0190   -0.0788    0.0833   -0.0150
    0.0308    0.0592    0.0656    0.1169    0.1527   -0.9769
    0.1231    0.1363    0.3542    0.6222    0.6398    0.2104
    0.3077    0.1902    0.4100    0.4161   -0.7264   -0.0150
    0.3385    0.4500    0.4961   -0.6366    0.1761    0.0225
R1 =
   32.4962   26.6801   21.4795   23.8182   26.0031
         0   19.9292   12.4403    2.1340    4.3271
         0         0   24.4514   11.8132    3.9931
         0         0         0   20.2382   10.3392
         0         0         0         0   16.1948
         0         0         0         0         0
Aqr =
   17.0000   24.0000    1.0000    8.0000   15.0000
   23.0000    5.0000    7.0000   14.0000   16.0000
    1.0000    2.0000    3.0000    4.0000    5.0000
    4.0000    6.0000   13.0000   20.0000   22.0000
   10.0000   12.0000   19.0000   21.0000    3.0000
   11.0000   18.0000   25.0000    2.0000    9.0000
A2 =
    17    24     1     8    15
    23     5     7    14    16
     1     2     3     4     5
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

从上面的结果中可以看出，在上面的结果中，满足等式$Aqr=Q_1 \cdot R_1$，其中Aqr就是矩阵A删除对应数据行的结果，也就是上面结果中的A_2矩阵。

step 3 证明Q_1矩阵的正交性。在MATLAB的命令窗口中输入下面的命令：

```
>> detQ1=det(Q1);
for i=1:5
A=Q1(:,i);
for j=(i+1):6
B=Q1(:,j);
C=A'*B;
disp(num2str(C))
end
end
```

step 4 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
detQ1 =
    1.0000
C=
-5.5511e-017
5.5511e-017
0
-6.9389e-018
```

```
-4.3368e-018
-5.5511e-017
0
-1.3878e-017
-1.7347e-018
0
0
-3.9899e-017
-2.6368e-016
-4.8572e-017
-3.9031e-017
```

奇异值分解

奇异值分解在矩阵分析中有着重要的地位，对于任意矩阵 $A \in C^{m\times n}$，存在酉矩阵（Unitray matrix），$U=[u_1, u_2, \cdots, u_n]$，$V=[v_1, v_2, \cdots, V_n]$。使得：

$$U^T AV = \mathrm{diag}(\sigma_1,\sigma_2,\cdots,\sigma_p)$$

其中参数 $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_p, p=\min\{m,n\}$。在上面的式子中，$\{\sigma_i,u_i,v_i\}$分别是矩阵$A$的第$i$个奇异值、左奇异值和右奇异值，它们的组合就称为奇异值分解三对组。

在 MATLAB 中，计算奇异值分解的命令如下：

- [U,S,V] = svd(X) 奇异值分解；
- [U,S,V] = svd(X,0) 比较经济的奇异值分解；
- s = svds(A,k,0) 向量 s 中包含矩阵 A 分解得到的 k 个最小奇异值；
- [U,S,V] = svds(A,k,0) 给出 A 的 k 个最大奇异值分解结果。

例 3.21 对矩阵进行奇异值分解。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> X=[ 1     2
       3     4
       5     6
       7     8] ;
>> [ U,S,V]  = svd(X)
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
U =
   -0.1525   -0.8226   -0.3945   -0.3800
   -0.3499   -0.4214    0.2428    0.8007
   -0.5474   -0.0201    0.6979   -0.4614
   -0.7448    0.3812   -0.5462    0.0407
S =
   14.2691         0
         0    0.6268
         0         0
         0         0
V =
   -0.6414    0.7672
   -0.7672   -0.6414
```

step 3 使用最经济的方法来进行分解。在 MATLAB 的命令窗口中输入下面的命令：

```
>> X=[ 1     2
       3     4
       5     6
       7     8];
>> [U,S,V] = svd(X,0)
```

step 4 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
U =
   -0.1525   -0.8226
   -0.3499   -0.4214
   -0.5474   -0.0201
   -0.7448    0.3812
S =
   14.2691         0
         0    0.6268
V =
   -0.6414    0.7672
   -0.7672   -0.6414
```

例 3.22 使用 svd 和 svds 命令对稀疏矩阵进行奇异值分解。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> load west0479
s = svd(full(west0479));
sl = svds(west0479,4);ss = svds(west0479,6,0);
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
sl =
  1.0e+005 *
    3.1895
    3.1725
    3.1695
    3.1685
ss =
 1.0e-004 *
    0.5616
    0.5169
    0.4505
    0.4020
    0.0424
    0.0098
```



在上面的程序结果中，sl 表示该稀疏矩阵的最大的 4 个奇异值，ss 则表示该稀疏矩阵的最小的 6 个奇异值。

step 3 绘制数据结果图形。在 MATLAB 的命令窗口中输入下面的命令：

```
>> plot(sl,'ro')
```

```
>> hold on
>> plot(s,'gp')
>> set(gca,'Xtick',[0:1:5])
>> set(gca,'Xlim',[0 4.5])
>> xlabel('n')
>> ylabel('The singular value')
```

step 4 查看求解的结果。输入上面的程序代码后，按"Enter"键，得到的结果如图3.6所示。

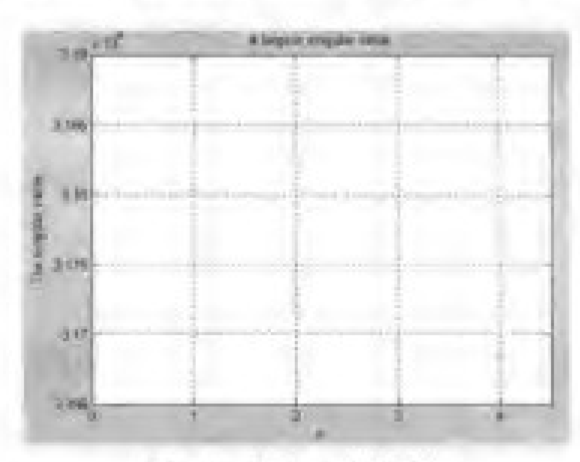


图3.6 最大4个奇异值



从上面的图形结果中可以看出，使用svd和svds命令求解的结果在某个误差容限之内，结果十分相近。

3.4 特征值分析

在线性代数的理论中，对于$n \times n$方阵$\boldsymbol{A}$，其特征值λ和特征向量$\boldsymbol{x}$满足下面的等式：

$$\boldsymbol{A}\boldsymbol{x}=\lambda \boldsymbol{x}$$

在上面的等式中，λ是一个标量，$\boldsymbol{x}$是一个向量。把矩阵$\boldsymbol{A}$的n个特征值放置在矩阵的对角线上就可以组成一个矩阵$\boldsymbol{D}$，也就是，$\boldsymbol{D}=\mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$，然后将各特征值对应的特征向量按照对应次序排列，作为矩阵$\boldsymbol{V}$的数据列。如果该矩阵$\boldsymbol{V}$是可逆的，则关于特征值的问题可以描述为：

$$\boldsymbol{A} \cdot \boldsymbol{V}=\boldsymbol{V} \cdot \boldsymbol{D} \Rightarrow \boldsymbol{A}=\boldsymbol{V} \cdot \boldsymbol{D} \cdot \boldsymbol{V}^{-1}$$

在MATLAB中，提供多种关于矩阵特征值处理的函数，用户可以使用这些函数来分析矩阵的特征值的多种内容，下面分小节详细分析。

3.4.1 特征值和特征向量

在MATLAB中，求解矩阵特征值和特征向量的数值运算方法为：对矩阵进行一系列的Householder变换，产生一个准上三角矩阵，然后使用QR法迭代进行对角化。对于一般读者来讲，可以不用了解这些计算原理。关于矩阵的特征值和特征向量的命令比较简单，具体的调用格式如下：

- d= eig(A) 仅计算矩阵$\boldsymbol{A}$的特征值，并且以向量的形式输出；
- [V,D] = eig(A) 计算矩阵$\boldsymbol{A}$的特征向量矩阵$\boldsymbol{V}$和特征值对角阵$\boldsymbol{D}$，满足等式$\boldsymbol{AV}=\boldsymbol{VD}$；

- [V,D] = eig(A,'nobalance') 当矩阵 **A** 中有截断误差数量级相差不大时，该指令更加精确；
- [V,D] = eig(A,B) 计算矩阵 **A** 的广义特征向量矩阵 **V** 和广义特征值对角阵 **D**，满足等式 **AV=BVD**；
- d = eigs(A,k,sigma) 计算稀疏矩阵 **A** 的 k 个由 sigma 指定的特征向量和特征值。关于参数 sigma 的取值，请查看相应的帮助文件。



当只需要了解矩阵的特征值的时候，推荐使用第一条命令，这样可以节约系统的资源，同时可以有效的得出结果。

例 3.23 对基础矩阵求解矩阵的特征值和特征向量。

step 1 对矩阵进行特征值分析。在 MATLAB 的命令窗口中输入下面的命令：

```
>> A=pascal(5);
>> [V D]=eig(A)
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
V =
    0.1680   -0.5706   -0.7660    0.2429    0.0175
   -0.5517    0.5587   -0.3830    0.4808    0.0749
    0.7025    0.2529    0.1642    0.6110    0.2055
   -0.4071   -0.5179    0.4377    0.4130    0.4515
    0.0900    0.1734   -0.2189   -0.4074    0.8649
D =
    0.0108         0         0         0         0
         0    0.1812         0         0         0
         0         0    1.0000         0         0
         0         0         0    5.5175         0
         0         0         0         0   92.2904
```

step 3 检验分析得到的结果。在 MATLAB 的命令窗口中输入下面的命令：

```
>> detV=det(V);
>> S=A*V-V*D;
```

step 4 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
detV =
    1.0000
S =
  1.0e-015 *
    0.0620   -0.0154    0.0304    0.1110         0
    0.1057   -0.0271    0.0694    0.0833         0
    0.0649   -0.0717    0.0900   -0.0555    0.1110
    0.0684   -0.0283    0.0416    0.1527    0.1110
    0.0753    0.0933   -0.0191    0.0555    0.1110
```



从上面的结果中可以看出，**V** 矩阵的行列式为 1，是可逆矩阵，同时求解得到的矩阵结果满足等式 **AV=VD**。

例 3.24 计算当矩阵的元素和截断误差相当的时候，矩阵的特征值和特征向量。

step 1 对矩阵进行特征值分析。在 MATLAB 的命令窗口中输入下面的命令：

```
>> B = [ 3      -4       -1.9    5*eps
     -2       3        2     -eps/5
     -eps/3  eps/2  -1       0
     -1.5    -.5      .1     1   ];
>> [ VB,DB] = eig(B);
>> ER1=B*VB - VB*DB;
>> [ VN,DN] = eig(B,'nobalance');
>> ER2=B*VN - VN*DN;
```



在上面的程序代码中，ER1 表示默认的误差，ER2 则表示经过 "nobalance" 参数处理后的误差。

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
VB =
    0.8016   -0.3920    0.0000    0.0420
   -0.5668   -0.2772    0.0000    0.4408
   -0.0000   -0.0000    0.0000   -0.8396
   -0.1903   -0.8772    1.0000   -0.3148
VN =
    1.0000   -0.4469   -0.0000    0.0500
   -0.7071   -0.3160   -0.0000    0.5250
   -0.0000   -0.0000   -0.0000   -1.0000
   -0.2374   -1.0000   -1.0000    0.2187

DB =
    5.8284         0         0         0
         0    0.1716         0         0
         0         0    1.0000         0
         0         0         0   -1.0000
DN =
    5.8284         0         0         0
         0    0.1716         0         0
         0         0    1.0000         0
         0         0         0   -1.0000

ER1 =
    0.0000    0.0000    0.0000   -0.0000
   -0.0000   -0.0000   -0.0000    0.0000
    0.0000   -0.0000    0.0000   -0.0000
   -0.0000   -0.0000   -0.0000   -0.9970
ER2 =
  1.0e-014 *
    0.1776    0.0569    0.0290    0.0701
    0.0888   -0.0430   -0.0173   -0.0333
    0.0006    0.0021    0.0020   -0.0222
         0   -0.0167         0         0
```





从上面的结果可以看出，如果矩阵中的元素和截断误差相当时，如果在命令中不选用 "nobalance" 参数，则求取的结果是有严重计算错误的。当矩阵 *A* 来自程序中的中间计算结果而且结果中包含 eps 元素时，就会发生上面的情况。

3.4.2 稀疏矩阵的特征值和特征向量

例 3.25 使用 eigs 命令来求取稀疏矩阵的特征值和特征向量。

step 1 生成稀疏矩阵，并求取特征值。在 MATLAB 的命令窗口中输入下面的命令：

```
>>A = delsq(numgrid('C',30));
>>d = eig(full(A));
>>[ dum,ind] = sort(abs(d));
>>dlm = eigs(A);
>>dsm = eigs(A,6,'sm');
>>dsmt=sort(dsm);
>>subplot(2,1,1)
>>plot(dlm,'r+')
>>hold on
>>plot(d(ind(end:-1:end-5)),'rs')
>>hold off
>>legend('eigs(A)','eig(full(A))',3)
>>set(gca,'XLim',[ 0.5 6.5] )
>>grid
>>title('Six largest magnitude eigenvalues')
>>subplot(2,1,2)
>>plot(dsmt,'r+')
>>hold on
>>plot(d(ind(1:6)),'rs')
>>hold off
>>legend('eigs(A,6,''sm'')','eig(full(A))',2)
>>grid
>>set(gca,'XLim',[ 0.5 6.5] )
>>title('Six samllest magnitude eigenvalues')
```

step 2 查看求解的结果。在输入上面的程序代码后，按 "Enter" 键，得到的图形如图 3.7 所示。

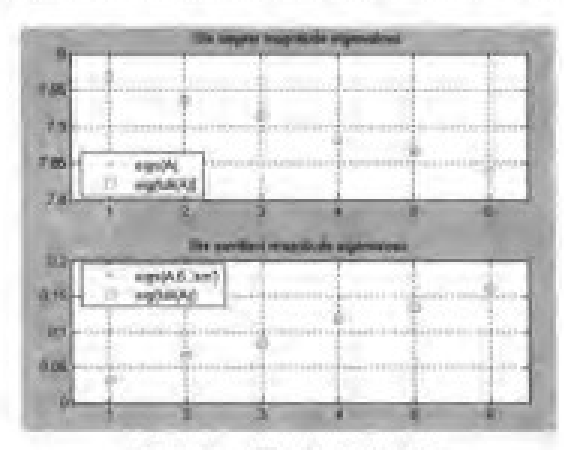


图 3.7 计算的图形结果



在本例中，矩阵 A 是一个对称的正定矩阵，维度是632，其对应的特征值均匀分布在(0,8)上，但是有特征值 4 重复出现了 18 次。

3.4.3 特征值问题的条件数

在前面的章节中介绍过，如果在MATLAB中求解代数方程的条件数，这个命令不能用来求解矩阵特征值对扰动的灵敏度。矩阵特征值条件数定义是对矩阵的每个特征值进行的，其具体的定义如下：

$$c_i = \frac{1}{\cos\theta(v_{li}, v_{ri})}$$

在上面的等式中 v_{li}, v_{ri}，分别是特征值 λ_i 所对应的左特征行向量和右特征列向量。其中 $\theta(\cdot,\cdot)$，表示的是两个向量的夹角。

在 MATLAB 中，计算特征值条件数的命令如下：

- c = condeig(A)　向量 c 中包含了矩阵 A 中关于各特征值的条件数
- [V,D,s] = condeig(A)　该命令相等于[V,D] = eig(A)和 c = condeig(A)的组合

例 3.26　使用命令分别求解方程组的条件数和特征值条件数。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>>A=magic(10);
>>cequ=cond(A);
>>ceig=condeig(A);;
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
cequ =
  2.0660e+018
ceig =
    1.0000
    3.0261
    4.9128
    4.7390
    1.0802
    2.8862
    1.4891
    2.3247
    2.3247
    3.6460
```



从上面的结果来看，方程的条件数很大，但是矩阵特征值的条件数则比较小，这就表明了方程的条件数和对应矩阵特征值条件数是不等的。

step 3 重新计算新的矩阵，进行分析。在 MATLAB 的命令窗口中输入下面的命令：

```
>> A=eye(5,5);
>> A(3,2)=1;
>>A(2,5)=1;
>> cequ=cond(A);
```

```
>>ceig=condeig(A);
```

step 4 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 2.465190e-032.
A =
     1     0     0     0     0
     0     1     0     0     1
     0     1     1     0     0
     0     0     0     1     0
     0     0     0     0     1
cequ =
    4.0489
ceig =
  1.0e+031 *
    0.0000
    0.0000
    2.0282
    0.0000
    2.0282
```



从上面的结果中可以看出，在上面的例子中方程组的条件数很小，而对应的特征值条件数则有两个分量相当大。

例 3.27 对亏损矩阵进行条件数分析。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> A=gallery(5);
>> [V,D,c_eig]=condeig(A)
>> condA=cond(A)
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
V =
  0.0000       -0.0000 + 0.0000i  -0.0000 - 0.0000i
                0.0000 + 0.0000i   0.0000 - 0.0000i
  0.0206        0.0206 + 0.0001i   0.0206 - 0.0001i
                0.0207 + 0.0001i   0.0207 - 0.0001i
  0.1398       -0.1397 + 0.0001i  -0.1397 - 0.0001i
               -0.1397 + 0.0000i  -0.1397 - 0.0000i
 -0.9574        0.9574             0.9574
                0.9574             0.9574
 -0.2519        0.2519 - 0.0000i   0.2519 + 0.0000i
                0.2519 - 0.0000i   0.2519 + 0.0000i

D =
   -0.0408              0                  0                  0                 0
         0     -0.0119 + 0.0386i           0                  0                 0
         0              0          -0.0119 - 0.0386i  0                         0
         0              0                  0           0.0323 + 0.0230i    0
         0              0                  0                  0 0.0323 - 0.0230i
c_eig =
```

```
  1.0e+010 *
     2.1293
     2.0796
     2.0796
     2.0020
     2.0020
condA =
  2.0253e+018
```



在上面的步骤中，矩阵A是代数重复度为5、几何重复度为1的亏损矩阵，这是一个十分特殊的矩阵，其方程组的条件数和特征值的条件数都很大。eig命令还不能适用于这类矩阵的，所求得的结果是不可信的。

3.4.4 特征值的复数问题

在理论上即使是实数矩阵，其对应的特征值也有可能是复数。在实际应用中，经常需要将一对共轭复数特征值转换为一个实数块，为此 MATLAB 提供了下面的命令：

- [VR,DR] = cdf2rdf(VC,DC) 把复数对角形转换成实数对角形
- [VC,DC] = rsf2csf(VR,DR) 把实数对角形转换成复数对角形



在上面命令的参数中，DC表示含有复数的特征值对角阵，VC表示对应的特征向量矩阵；DR表示含有实数的特征值对角阵，VR表示对应的特征向量矩阵。

例 3.28 对矩阵的复数特征值进行分析。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> X=[ 1      2      3
       0      4      5
       0     -5      4];
>> [ VC, DC] = eig(X);
>> [ VR, DR] = cdf2rdf(VC,DC);
>> XR=VR* DR/VR;
>> XC=VC* DC/VC;
```

step 2 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
VC =
   1.0000              -0.0191 - 0.4002i  -0.0191 + 0.4002i
        0                    0 - 0.6479i        0 + 0.6479i
        0               0.6479              0.6479
DC =
   1.0000                    0                  0
        0               4.0000 + 5.0000i        0
        0                    0              4.0000 - 5.0000i
VR =
    1.0000   -0.0191   -0.4002
         0         0   -0.6479
         0    0.6479         0
DR =
```

```
     1.0000          0          0
          0     4.0000     5.0000
          0    -5.0000     4.0000
XC =
   1.0000              2.0000 + 0.0000i    3.0000
        0              4.0000              5.0000
        0             -5.0000              4.0000
XR =
     1.0000     2.0000     3.0000
          0     4.0000     5.0000
          0    -5.0000     4.0000
```

函数的零点

对于某任意函数，在求解范围之内可能有零点，也可能没有零点；可能只有一个零点，也可能有多个甚至无数个零点。因此，这就给程序求解函数的零点增加了很大的难度，没有可以求解所有函数零点的通用求解命令。本节将简单讨论一元函数和多元函数的零点求解问题。



一元函数的零点

在所有函数中，一元函数是最简单的，同时也是可以使用MATLAB提供的图形绘制命令来实现可视化的。因此，在本小节中将首先讨论一元函数零点的求取方法。

在 MATLAB 中，求解一元函数零点的命令是 fzero，其调用格式如下：

- x = fzero(fun,x0)　参数 fun 表示的是一元函数，x0 表示求解的初始数值；
- [x,fval,exitflag,output] = fzero(fun,x0,options)：参数options的含义是指优化迭代所采用的参数选项，该参数和后面章节中需要讲解到的 fsolve、fminbnd、fminsearch 等命令中的 options 都是相同的“模块”；在输出参数中，fval 表示对应的函数值，exitflag 表示程序退出的类型，output 则反映优化信息的变量。

例 3.29　求函数 $f(x)=x^2\sin x-x+1$ 在数值区间[-3,4]中的零点。

step 1 绘制函数的图形。在 MATLAB 的命令窗口中输入下面的命令：

```
%计算函数数值
>> x=[ -3:0.1:4] ;
>>y=sin(x).*x.^2-x+1;
%绘制函数图形
>>plot(x,y,'r','LineWidth',1.5)
>>hold on
%添加水平线
>>h=line([ -3,4] ,[ 0,0] );
%设置直线的宽度和颜色
>>set(h,'LineWidth',1.5)
>>set(h,'color','k')
%设置坐标轴刻度
>>set(gca,'Xtick',[ -3:0.5:4] )
%添加图形标题和坐标轴名称
>>title('The zero of function')
>> grid
>> xlabel('x')
```

```
>> ylabel('f(x)')
```

step 2 查看图形。输入上面的程序代码后，按"Enter"键，得到的图形如图 3.8 所示。

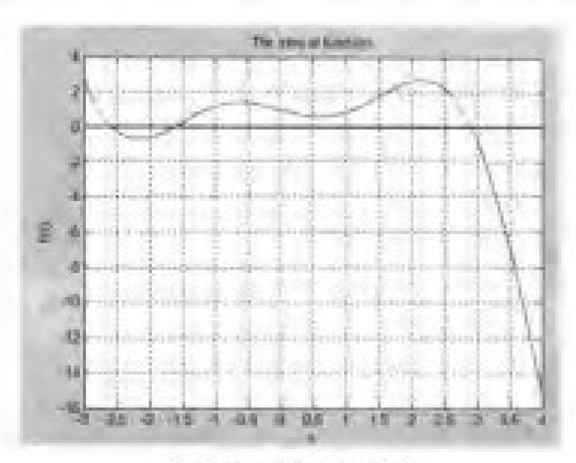


图 3.8 函数的图形



说明 之所以在求解函数零点之前，需要绘制函数的图形，是为了在后面的步骤中使用 fzero 命令时，更好地选择初始数值 x0。

step 3 求解函数的零点。在 MATLAB 的命令窗口中输入下面的命令：

```
>> [x1,f1,exitflag1]=fzero(f,-2.5);
>>[x2,f2,exitflag2]=fzero(f,-1.5);
>>[x3,f3,exitflag3]=fzero(f,3);
>> x=[x1,x2,x3];
>> f=[f1,f2,f3];
```

step 4 查看求解的结果。在命令窗口中输入计算的变量名称，得到的结果如下：

```
x =
   -2.5708   -1.6194    2.9142
f =
  1.0e-015 *
   -0.8882    0.2220   -0.8882
```

从上面的结果中可以看出，函数 $f(x)=x^2\sin x-x+1$ 在[-3,4]范围内的三个零点数值解为 -2.5708、-1.6194 和 2.9142。



提示 对于一元多项式函数，MATLAB 提供 roots 命令来求解多项式函数的所有零点，该命令的基本原理是求解多项式伴随矩阵的特征值来求解的。

3.5.2 多元函数的零点

一般来讲，多元函数的零点问题比一元函数的零点问题更难解决，但是当零点大致位置和性质比较好预测时，也可以使用数值方法来搜索精确的零点。

在 MATLAB 中，求解多元函数的命令是 fsolve，其具体的调用格式如下：

- x = fsolve(fun,x0)　解非线性方程组的数值解；
- [x,fval,exitflag,output] = fsolve(fun,x0,options)　完整格式，参数含义可以参考上面的函数。

例 3.30　求二元方程组 $\begin{cases}2x_1 - x_2 = e^{-x_1} \\ -x_1 + 2x_2 = e^{-x_2}\end{cases}$ 的零点。

step 1 绘制函数的图形。在 MATLAB 的命令窗口中输入下面的命令：

```
% 创建三维图形的数据网格
x=[-5:0.1:5];
y=x;
[X,Y]=meshgrid(x,y);
% 计算三维函数的数值
Z=2*X-Y-exp(-1*X);
% 绘制曲面图
surf(X,Y,Z)
% 设置照明属性
shading interp
% 添加水平的颜色条
colorbar horiz
% 设置图形的坐标轴刻度属性
set(gca,'Ztick',[-180:20:20])
set(gca,'ZLim',[-170 20])
% 设置透明属性
alphamap('rampdown')
colormap hot
% 添加图形标题和坐标轴名称
title('The figure of the function')
xlabel('x')
ylabel('y')
zlabel('z')
```

step 2 查看图形。在输入上面的代码后，按"Enter"键，得到函数图形如图 3.9 所示。

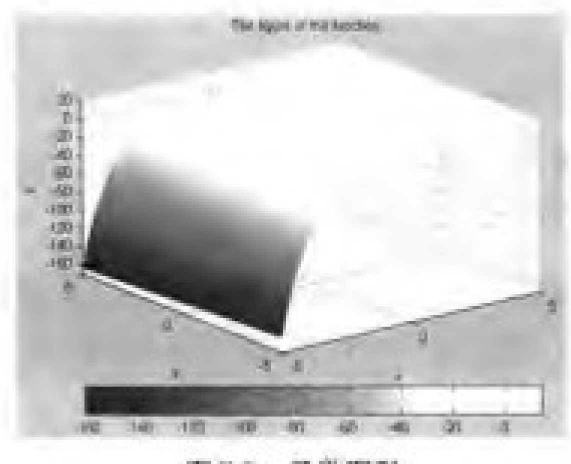


图 3.9　函数图形

step 3 编写求解函数的 M 文件。选择命令窗口编辑栏中的"File" ⇨ "New" ⇨ "M-File"命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function F = fsolvefun (x)
F = [ 2*x(1) - x(2) - exp(-x(1));
      -x(1) + 2*x(2) - exp(-x(2))];
```

在输入上面的程序代码后，将代码保存为"fsolvefun.m"文件。

step 4 求解二元函数的零点。在MATLAB的命令窗口中输入下面的代码：

```
x0 = [ -5; -5] ;
options=optimset('Display','iter');
[ x,fval] = fsolve(@fsolvefun,x0,options)
```

step 5 查看求解结果。在输入上面的代码后，按"Enter"键，得到下面的结果：

```
                                  Norm of      First-order   Trust-region
Iteration  Func-count     f(x)          step         optimality    radius
     0          3         47071.2                    2.29e+004         1
     1          6         12003.4              1     5.75e+003         1
     2          9         3147.02              1     1.47e+003         1
     3         12         854.452              1           388         1
     4         15         239.527              1           107         1
     5         18         67.0412              1          30.8         1
     6         21         16.7042              1          9.05         1
     7         24         2.42788              1          2.26         1
     8         27        0.032658       0.759511         0.206       2.5
     9         30    7.03149e-006       0.111927       0.00294       2.5
    10         33    3.29525e-013     0.00169132     6.36e-007       2.5
Optimization terminated: first-order optimality is less than options.
TolFun.
x =
    0.5671
    0.5671
fval =
  1.0e-006 *
   -0.4059
   -0.4059
```



从上面的结果中可以看出，由于原来的二元函数是对称的，因此所求解的未知数结果是相等的。由于在上面的实例中设置了显示迭代，因此在上面的结果中显示各优化信息。

3.6 数值积分

微积分是高等数学的重要知识，在工程实践中，微积分有着十分广泛的应用，因此如何通过计算机实现微积分是十分重要的内容。在MATLAB中，用户可以使用多种方法来实现微积分的运算：数值积分、符号积分、样条积分和Simulink模拟积分等。在本章中，将主要介绍数值积分和样条积分，并辅以介绍符号积分和Simulink积分等方法。

3.6.1 一元函数的数值积分

在MATLAB中，对一元函数进行数值积分的命令是quad和quadl。一般来讲，quadl命令比quad命令更加有效，它们的主要功能在于计算闭型数值积分，其对应的详细调用格式如下：

- q = quad(fun,a,b,tol,trace) 采用递推自适应Simpson法计算积分。
- q = quadl(fun,a,b,tol,trace) 采用递推自适应Lobatto法来计算积分。

下面详细介绍上面函数的参数含义。

- fun：被积函数，可以是字符串、内联函数、M函数文件名称的函数句柄。被积函数中一般使用x来作为自变量。
- a、b：被积函数的上限和下限，必须都是确定的数值。
- tol：标量，控制绝对误差，默认的数值精度是10^{-6}。
- trace：如果该输入变量的数值不是零，则随积分的进程逐点绘制被积函数。



在更为完整的调用命令q = quadl(fun,a,b,tol,trace,p1,p2...)中，p1和p2表示通过程序向被积函数传递的参数。

例3.31 求解积分 $\int_0^{3\pi}\sqrt{4\cos(2t)^2+\sin(t)^2+1}dt$ 的数值。

step 1 分析参数方程。根据微积分的基础知识，上面积分的数值实际上是某曲线的长度，该函数对应的参数方程如下：

$$\begin{cases} x(t)=\sin(2t) \\ y(t)=\cos(t) \\ z(t)=t \end{cases}$$

step 2 绘制函数图形。为了让读者了解该积分的数学含义，用户可以绘制该函数的图形。在MATLAB的命令窗口中输入下面的程序代码：

```
%绘制函数图形
t = 0:0.1:3*pi;
h=plot3(sin(2*t),cos(t),t,'r');
set(h,'LineWidth',1.5)
grid on
```

step 3 查看函数图形。在输入上面的程序代码后，按"Enter"键，得到的函数图形如图3.10所示。

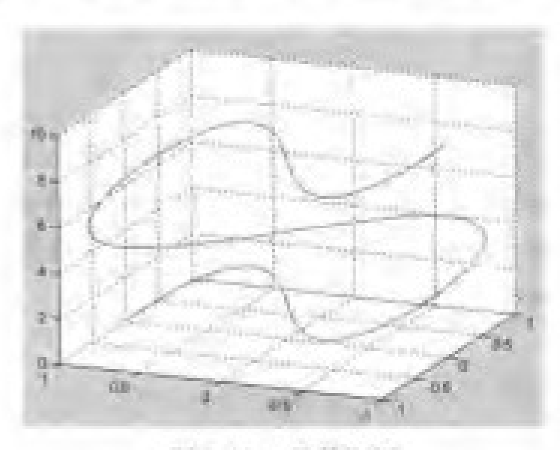


图3.10 函数图形

step 4 编写被积函数的M文件。选择命令窗口编辑栏中的"File" ⇒ "New" ⇒ "M-File"命令，打开M文件编辑器，在其中输入下面的程序代码：

```
%编写被积函数的M文件
function f = hcurve(t)
f = sqrt(4*cos(2*t).^2 + sin(t).^2 + 1);
```

在输入上面的程序代码后，将代码保存为"hcurve.m"文件。

step 5 使用quad和quadl命令来求解数值积分。在命令窗口中输入下面的程序代码：

```
>> len1=quad(@hcurve,0,3*pi);
>> len2=quadl(@hcurve,0,3*pi);
```

step 6 查看求解结果。在命令窗口中输入变量名称，得到求解的结果如下：

```
len1 =
   17.2220
len2 =
   17.2220
```

step 7 设置积分的求解属性，重新求解数值积分。在命令窗口中输入下面的程序代码：

```
>>len3=quad(@hcurve,0,3*pi,1.e-3,1);
>>len4=quadl(@hcurve,0,3*pi,1.e-3,1);
```

step 8 查看求解结果。在输入上面的代码后，按"Enter"键，得到下面的结果：

```
      9     0.0000000000     2.55958120e+000     4.5159602105
     11     0.0000000000     1.27979060e+000     2.1975964146
     13     0.0000000000     6.39895300e-001     1.1908151623
     15     0.6398952996     6.39895300e-001     0.9939908843
     ........................................//限于篇幅，省略了部分数据
     53     8.1449873615     1.27979060e+000     2.1975964146
     55     8.1449873615     6.39895300e-001     0.9939908843
     57     8.7848826611     6.39895300e-001     1.1908151623
len3 =
   17.2099
     18     0.0000000000     4.71238898e+000     15.8755795718
     23     0.0000000000     4.32369745e-001     1.4707654142
     28     0.0000000000     3.96706633e-002     0.1768555623
     33     0.0793413265     7.98333951e-002     0.3426629563
     ......................................... //限于篇幅，省略了部分数据
    208     8.6393797974     7.98333951e-002     0.1989870400
    213     8.7990465876     9.66808141e-002     0.2889072894
    218     8.9924082158     9.66808141e-002     0.3638528604
    223     9.1857698440     7.98333951e-002     0.3426629563
    228     9.3454366343     3.96706633e-002     0.1768555623
len4 =
   17.2220
```



从上面的结果可以看出，当用户设置比较小的误差后，使用quad和quadl命令求解得到的结果精度有很大的差别。

3.6.2 使用Simulink求解数值积分

例3.32 使用Simulink求解积分$\int_0^{3\pi}\sqrt{4\cos(2t)^2+\sin(t)^2+1}dt$的数值。

step 1 选择MATLAB编辑栏中的"File"⇨"New"⇨"Model"命令，打开模型编辑器，向其中添加对应的模型块，如图3.11所示。

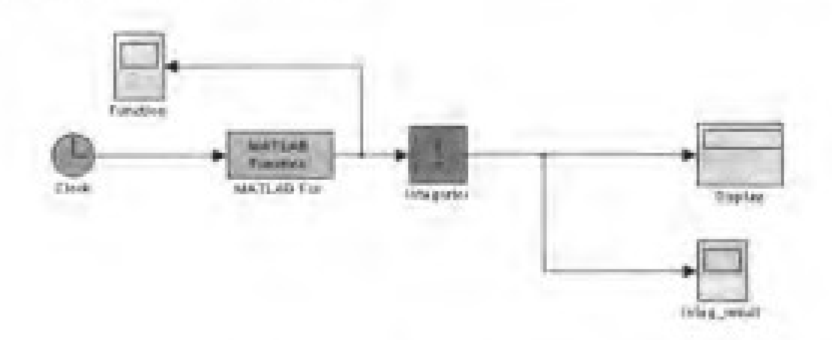


图3.11 添加系统的模块

step 2 设置系统模块的属性。双击上面系统中的"MATLAB Fcn"模块，打开对应的模块属性对话框，在其中设置被积函数表达式，如图3.12所示。



图3.12 设置系统模块的属性

step 3 设置系统仿真的时间，运行系统仿真。将系统仿真的时间设置为3π，然后运行仿真，得到的结果如图3.13所示。

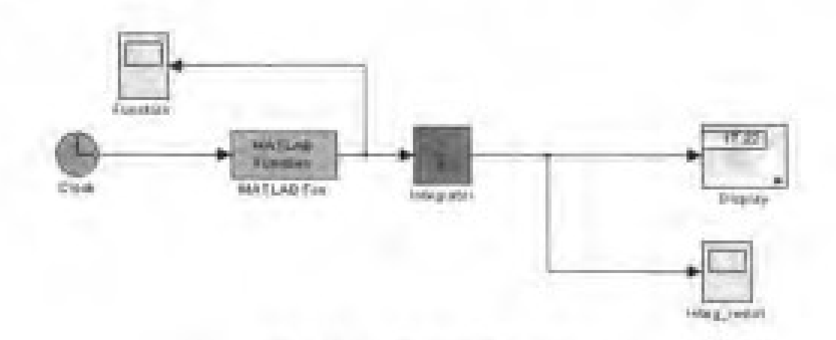


图3.13 查看仿真的结果

step 4 查看被积函数的图形。从上面的图形中可以看出，通过Simulink积分得到的结果是17.22。

同时，用户可以查看被积函数的图形，如图 3.14 所示。

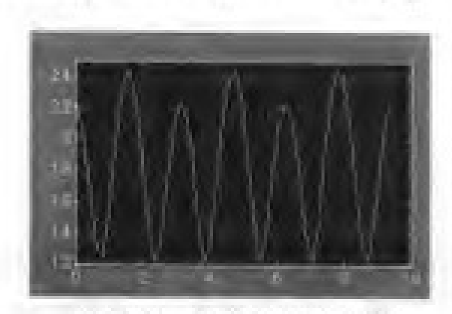


图 3.14　被积函数的图形



关于 Simulink 中如何添加对应的系统模块，如何设置模块属性，请用户查看本书中关于 Simulink 的相关章节的内容。

3.6.3 求解瑕积分

例 3.33　求解积分 $\int_0^1 \sqrt{\ln - \frac{1}{x}} dx$ 的数值。

step 1 分析积分的问题。在上面的积分表达式中，由于在 $x=0$ 时，被积函数 $\sqrt{\ln \frac{1}{x}}$ 的数值趋近于无穷大，也就是 $\lim_{x \to 0^+} \sqrt{\ln \frac{1}{x}} = +\infty$。因此，上面的积分属于瑕积分或者称为开型积分，这和上面的有限积分在性质上有比较大的差异。同时根据基础的高等数学知识，该积分数值为 $\int_0^1 \sqrt{\ln \frac{1}{x}} dx = \frac{\sqrt{\pi}}{2} \approx 0.8862$。

step 2 求解积分数值。在 MATLAB 的命令窗口中输入下面的代码：

```
>>f=inline('sqrt(log(1./x))','x');
>> Infq=quad(f,0,1);
>> Infql=quadl(f,0,1);
```

step 3 查看求解结果。在命令窗口中输入变量名称，得到求解的结果如下：

```
Warning: Divide by zero.
> In inlineeval at 13
  In inline.subsref at 25
  In quad at 62
Infq =
    0.8862
Warning: Divide by zero.
> In inlineeval at 13
  In inline.feval at 34
  In quadl at 64
Infql =
    0.8862
```

step 4 使用符号方法来求解。在 MATLAB 的命令窗口中输入下面的代码：

```
>> f=inline('sqrt(log(1/x))','x');
>> syms x
>> Is=vpa(int('sqrt(log(1/x))','x',0,1))
```

step 5 查看求解结果。在命令窗口中输入变量名称，得到求解的结果如下：

```
Warning: Explicit integral could not be found.
> In sym.int at 58
  In char.int at 9
 Is =
.88622692545275801364908374167057
```



从上面的结果可以看出，使用符号方法求得得到的结果和数值积分得到的结果相同，精度都是可以接受的。但是，符号运算比数值运算更加占用系统资源。

3.6.4 矩形区域的多重数值积分

多重数值积分可以认为是一元函数积分的推广和延伸，但是情况比一元函数要复杂，在本小节中将主要介绍如何在 MATLAB 中计算二重数值积分。

在 MATLAB 中，计算二重数值积分的命令为 dblquad，其具体的调用格式如下：

```
q = dblquad(fun,xmin,xmax,ymin,ymax,tol,method)
```

在上面参数中，fun 表示积分函数，xmin,xmax 表示变量 x 的上、下限，ymin,ymax 表示变量 y 的上、下限；tol 表示积分绝对误差，默认数值为 10^{-6}；method 表示积分方法的选项，默认选项为 @quad，用户可以选择 @quadl 或者自己定义的积分函数句柄。

例 3.34 求解积分 $\int_0^\pi\int_\pi^{2\pi}(y\sin x + x\cos y)\mathrm{d}x\mathrm{d}y$ 的数值。

step 1 求解积分数值。在 MATLAB 的命令窗口中输入下面的代码：

```
>>integrnd=@(x,y) y*sin(x)+x*cos(y);
>>xmin = pi;
>>xmax = 2*pi;
>>ymin = 0;
>>ymax = pi;
>>result = dblquad(integrnd,xmin,xmax,ymin,ymax)
```

step 2 查看求解结果。输入上面的代码后，按 "Enter" 键，得到求解的结果如下：

```
result =
   -9.8696
```

step 3 使用符号运算求解积分数值。

```
>>syms x y
>> result1=vpa(int(int((y*sin(x)+x*cos(y)),x,pi,2*pi),y,0,pi))
```

step 4 查看求解结果。输入上面的代码后，按 "Enter" 键，得到求解的结果如下：

```
result1 =
 -9.8696044010893586188344909998761
```



变量区域的多重数值积分

前面所介绍的内容中，都是固定数值的二重积分运算方法，但是在实际应用中，二重积分并不都是矩形计算区域，在计算区域中会包含变量表达式。也就是说，积分区域可以表示成下面的情况：

$$R=\{(x,y)\,|\,a\leqslant x\leqslant b,c(x)\leqslant y\leqslant d(x)\}$$

用户需要求解的积分表达式为：

$$I=\int\int_R f(x,y)\mathrm{d}x\mathrm{d}y=\int_a^b\left\{\int_{c(x)}^{d(x)}f(x,y)\mathrm{d}y\right\}\mathrm{d}x$$

对于上面的积分表达式，进行数值计算的表达式为：

$$I(a,b,c(x),d(x))=\sum_{m=1}^{M}w_m\sum_{n=1}^{N}v_n f(x_m,y_{m,n})$$

在上面的表达式中 w_m、v_n 表示的是权重，取决于一维积分方法。关于上面二重积分的数值分析的方法，如图 3.15 所示。

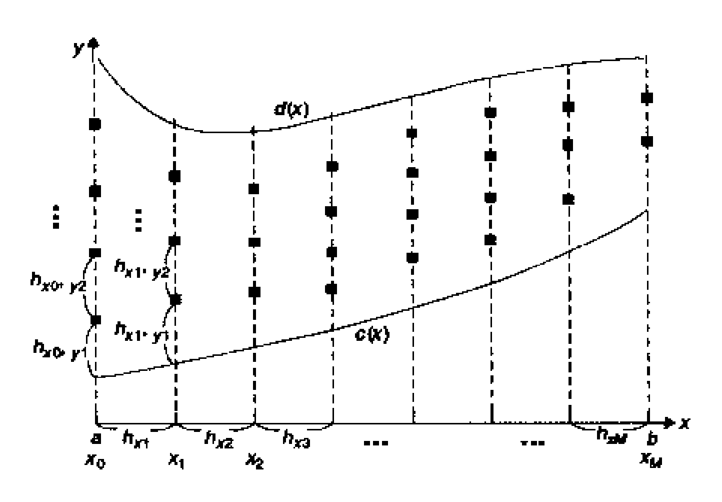


图 3.15　二重积分数据点

根据上面的数据点区域，用户需要自行编写M文件来计算上面的数值积分。在本小节中，将使用一个简单的例子说明如何计算二重数值积分。

例 3.35　求解积分 $I=\int_{-1}^{1}\int_{0}^{\sqrt{1-x^2}}\sqrt{1-x^2-y^2}\mathrm{d}y\mathrm{d}x$ 的数值。

step 1 编写一维数值积分的 M 文件。选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function INTf=smpsns_fxy(f,x,c,d,N)
% 函数 f(x,y)的一维数值积分数值;
% 对应的积分区域是 Ry={ c<=y<=d}

% 当用户没有输入函数中的 N 参数时，默认值为 100
if nargin<5
    N=100;
end

% 当参数 c=d 或者参数 N=0 时，返回积分数值为 0
if abs(d-c)<eps|N<=0
    INTf=0;
```

```
    return;
end
% 如果参数N是奇数，则将其加1，变成偶数
if mod(N,2)~=0
    N=N+1;
end
% 计算单位高度数值
h=(d-c)/N;
% 计算节点的y轴坐标值
y=c+[0:N]*h;
% 计算节点的积分函数数值
fxy=feval(f,x,y);
% 确定积分的限制范围
fxy(find(fxy==inf))=realmax;
fxy(find(fxy==-inf))=-realmax;
% 计算奇数和偶数的节点x坐标数值
kodd=2:2:N;
keven=3:2:N-1;
% 根据积分公式得出积分数值
INTf=h/3*(fxy(1)+fxy(N+1)+4*sum(fxy(kodd))+2*sum(fxy(keven)));
```

在输入上面的程序代码后，将代码保存为"smpsns_fxy.m"文件。

step 2 编写二重数值积分的M文件。选择命令窗口编辑栏中的"File" ➪ "New" ➪ "M-File"命令，打开M文件编辑器，在其中输入下面的程序代码：

```
function INTfxy=int2s(f,a,b,c,d,M,N)
% 被积函数f(x,y)的二重积分数值
% 积分区域为R ={(x,y)|a<=x<=b,c(x)<=y<=d(x)}
% 使用的积分方法是Simpson法则

if ceil(M)~=floor(M)
    hx=M;
    M=ceil((b-a)/hx);
end
if mod(M,2)~=0
    M=M+1;
end
hx=(b-a)/M;
m=1:M+1;
x=a+(m-1)*hx;

% 判断参数c是否是数值
% 如果c是数值，将积分限制设置为数值c
% 如果c不是数值，则将积分显示设置为函数表达式
if isnumeric(c)
    cx(m)=c;
else
    cx(m)=feval(c,x(m));
end

% 判断参数d是否是数值
% 如果d是数值，将积分限制设置为数值c
% 如果d不是数值，则将积分显示设置为积分表达式
if isnumeric(d)
    dx(m)=d;
```

```
else
    dx(m)=feval(d,x(m));
end

%重复和参数M类似的操作
if ceil(N)~=floor(N)
    hy=N;
    Nx(m)=ceil((dx(m)-cx(m))/hy);
    ind=find(mod(Nx(m),2)~=0);
    Nx(ind)=Nx(ind)+1;
else
    if mod(N,2)~=0
        N=N+1;
    end
    Nx(m)=N;
end

%根据Simpson法则计算各个节点的数值
for m=1:M+1
    sx(m)=smpsns_fxy(f,x(m),cx(m),dx(m),Nx(m));
end

%计算奇数和偶数的节点
kodd=2:2:M;
keven=3:2:M-1;
%计算积分数值
INTfxy=hx/3*(sx(1)+sx(M+1)+4*sum(sx(kodd))+2*sum(sx(keven)));
```

在输入上面的程序代码后，将代码保存为“int2s.m”文件。

step 3 进行二重积分计算。在MATLAB的命令窗口中输入下面的代码：

```
>> x=[-1:0.05:1];
>> y=[0:0.05:1];
>>[X,Y]=meshgrid(x,y);
>>f510=inline('sqrt(max(1-x.*x-y.*y,0))','x','y');
>> Z=f510(X,Y);
>> d=inline('sqrt(max(1-x.*x,0))','x');
>> b=1;
>> a=-1;
>> c=0;
>> Vs1=int2s(f510,a,b,c,d,100,100);
>> error1=Vs1-pi/3;
>> Vs2=int2s(f510,a,b,c,d,0.01,0.01);
>> error2=Vs2-pi/3;
```

step 4 查看求解结果。在命令窗口中输入变量名称，然后按“Enter”键，得到求解的结果如下：

```
>> Vs1
Vs1 =
    1.0470
>> Vs2
Vs2 =
    1.0470
>> error1
error1 =
```

```
 -1.5315e-004
>> error2
error2 =
-1.9685e-004
```



在上面的程序结果中，Vs1 和Vs2是分别通过不同的计算方法得到的结果，error1 和 error2 是计算结果和真实结果之间的误差。

step 5 绘制函数图形。在 MATLAB 的命令窗口中输入下面的代码：

```
>> surf(X,Y,Z)
>>shading interp
>> colormap hsv
>> colorbar horiz
```

step 6 查看图形。输入上面的程序代码后，按"Enter"键，得到的结果如图 3.16 所示。

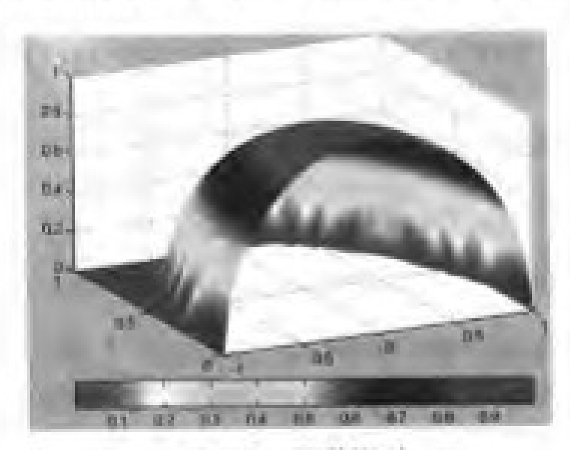


图 3.16　函数图形

3.7　概率论和数理统计

本节将主要介绍在MATLAB中运用概率论和数理统计的方法，主要的内容包括概率分布、数理统计和假设检验等。在每个具体的小节中，将主要介绍如何在MATLAB中运用相关知识，对于具体的背景知识，请读者查看对应的书籍。

3.7.1　双变量的概率分布

概率分布是概率论和数理统计的基础知识。在 MATLAB 中，提供处理常见概率分布的各种命令，包括二项分布、泊松分布、χ^2分布、t分布等概率分布。这些内容比较简单，在本小节就不详细展开介绍了，感兴趣的读者可以查阅相应的帮助文件。

在本小节中，将主要介绍如何在MATLAB中处理双变量或者多变量的概率分布的情况。首先介绍如何处理双变量t分布（bivariate t distribution）。根据基础的概率知识，描述双变量t分布的重要参数是线性相关矩阵k和自由度n。下面举例说明如何在MATLAB中显示多元分布的图形。

例 3.36　在 MATLAB 中使用图形来显示双变量t分布，其中两个变量服从的分布分别为：$t(1)$和$t(5)$，也就是说，两个变量的自由度分别为 1 和 5。下面使用图形显示在两个变量线性相关距阵k

的不同取值下的分布情况。

step 1 绘制二元概率分布的图形。在 MATLAB 的命令窗口中输入下面的代码：

```
% 设置分布参数
%n 代表数据点个数
%nu 表示自由度
% 相关系数矩阵为[ 1 .8; .8 1]
n = 500;
nu = 1;
% 产生多元 t 分布的随机数值矩阵
T = mvtrnd([ 1 .8; .8 1] , nu, n);
% 计算 t 分布数值的累积概率分布数值
U = tcdf(T,nu);

% 绘制数据点的图形，并设置图形的属性
subplot(2,2,1);
plot(U(:,1),U(:,2),'r.');
title('rho = 0.8');
xlabel('U1');
ylabel('U2')

% 相关系数矩阵为[ 1 .1; .1 1]
T = mvtrnd([ 1 .1; .1 1] , nu, n);
U = tcdf(T,nu);
% 绘制数据点的图形，并设置图形的属性
subplot(2,2,2);
plot(U(:,1),U(:,2),'r.');
title('rho = 0.1');
xlabel('U1');
ylabel('U2');

% 相关系数矩阵为[ 1 -.1; -.1 1]
T = mvtrnd([ 1 -.1; -.1 1] , nu, n);
U = tcdf(T,nu);
% 绘制数据点的图形，并设置图形的属性
subplot(2,2,3);
plot(U(:,1),U(:,2),'r.');
title('rho = -0.1');
xlabel('U1');
ylabel('U2');

% 相关系数矩阵为[ 1 -.8; -.8 1]
T = mvtrnd([ 1 -.8; -.8 1] , nu, n);
U = tcdf(T,nu);
% 绘制数据点的图形，并设置图形的属性
subplot(2,2,4);
plot(U(:,1),U(:,2),'r.');
title('rho = -0.8');
xlabel('U1');
ylabel('U2')
```

step 1 查看图形。在输入程序代码后，按“Enter”键，得到的图形如图 3.17 所示。

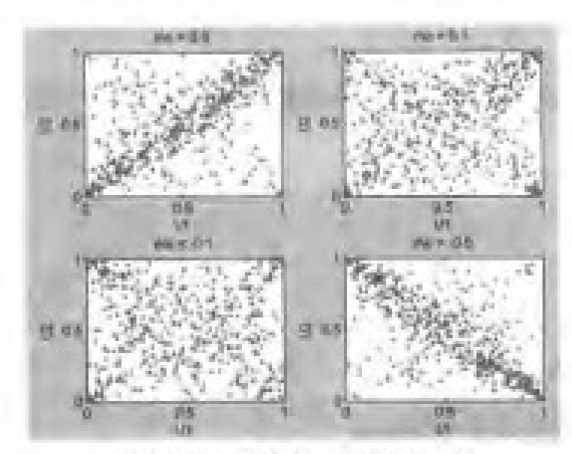


图 3.17　双变量 t 分布的图形

关于上面的程序代码，相关说明如下：

- 在上面的程序代码中，mvtrnd 命令的功能是从多元 t 分布中产生随机数据矩阵，关于其具体的用法，读者可以查看对应的帮助文件。
- tcdf 命令的功能是产生 t 分布的累积概率数值，具体的用法请查阅相应的帮助文件。



从上面的图形中可以看出，当两个变量之间的线性相关矩阵系数值的绝对值越接近 1，两个变量的分布更加紧密；而当两个变量之间的线性相关矩阵系数值的绝对值越接近 0，两个变量的分布更加稀疏。

3.7.2 不同概率分布

在MATLAB中，除了绘制两个相同分布变量之外，还可以绘制两个不同随机分布的变量的数据分布图，下面举例详细说明。

例 3.37　两个相关随机变量，分别服从 Gamma 分布和 t 分布，两个变量相互独立，且具体的随机变量参数为 Gamma(2,1)和 t(5)，在 MATLAB 中绘制两个变量的数据分布图形。

step 1 绘制二元概率分布的图形。在 MATLAB 的命令窗口中输入下面的代码：

```
subplot(1,1,1);

%设置概率分布的参数
n = 1000;
rho = .7;
nu = 1;

%产生多元 t 分布的随机数值矩阵
T = mvtrnd([1 rho; rho 1], nu, n);
%计算 t 分布数值的累积概率分布数值
U = tcdf(T,nu);
%产生两个概率分布的数值
X = [gaminv(U(:,1),2,1) tinv(U(:,2),5)];

%计算两个直方图的数值
[n1,ctr1] = hist(X(:,1),20);
[n2,ctr2] = hist(X(:,2),20);
```

```
%绘制图形
subplot(2,2,2);
plot(X(:,1),X(:,2),'.');
axis([0 15 -10 10]);
h1 = gca;
title('1000 Simulated Dependent t and Gamma Values');
xlabel('X1 ~ Gamma(2,1)');
ylabel('X2 ~ t(5)');
subplot(2,2,4); bar(ctr1,-n1,1);
axis([0 15 -max(n1)*1.1 0]);
axis('off');
h2 = gca;
subplot(2,2,1); barh(ctr2,-n2,1);
axis([-max(n2)*1.1 0 -10 10]);
axis('off');
h3 = gca;
set(h1,'Position',[0.35 0.35 0.55 0.55]);
set(h2,'Position',[.35 .1 .55 .15]);
set(h3,'Position',[.1 .35 .15 .55]);
colormap([.8 .8 1]);
```

step 2 查看图形。在输入程序代码后，按“Enter”键，得到的图形如图 3.18 所示。

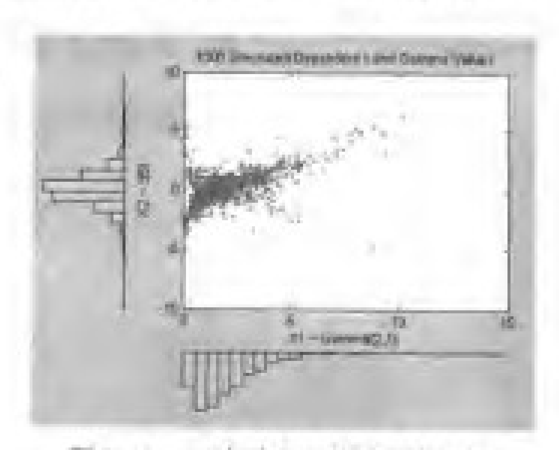


图 3.18 两个独立随机变量的图形

3.7.3 数据分布分析

在实际应用中，用户经常需要根据有限的试验数据，推测该样本数据所满足的数据分布情况，在本小节中，将使用一个简单的实例来演示如何在 MATLAB 中推测数据分布情况。

例 3.38 通过命令产生满足 t 分布的多元变量，然后使用自定义的概率密度函数来推测两个变量满足的多元正态分布 $N(\mu_1,\mu_2,\sigma_1,\sigma_2)$，其中参数分别表示均值和标准方差，然后绘制图形来验证这种推测是否正确。

step 1 绘制二元概率分布的图形。在 MATLAB 的命令窗口中输入下面的代码：

```
%产生随机数据
x = [trnd(20,1,50) trnd(4,1,100)+3];
%设置混合概率密度函数
pdf_normmixture = @(x,p,mu1,mu2,sigma1,sigma2) ...
                    p*normpdf(x,mu1,sigma1) + (1-p)*normpdf(x,mu2,sigma2);
```

```
%设置参数的数值
pStart = .5;
muStart = quantile(x,[.25 .75]);
sigmaStart = sqrt(var(x) - .25*diff(muStart).^2);
start = [pStart muStart sigmaStart sigmaStart];
%设置参数的上下限
lb = [0 -Inf -Inf 0 0];
ub = [1 Inf Inf Inf Inf];
%设置求解的属性
options = statset('MaxIter',300, 'MaxFunEvals',600);
paramEsts = mle(x, 'pdf',pdf_normmixture, 'start',start, ...
                        'lower',lb, 'upper',ub, 'options',options);
%绘制基础数据的直方图
bins = -2.5:.5:7.5;
h = bar(bins,histc(x,bins)/(length(x)*.5),'histc');
set(h,'FaceColor',[.9 .9 .9]);
xgrid = linspace(1.1*min(x),1.1*max(x),200);
%绘制概率密度图形
pdfgrid = pdf_normmixture(xgrid,paramEsts(1),paramEsts(2),paramEsts(3),
paramEsts(4),paramEsts(5));
hold on; plot(xgrid,pdfgrid,'-');
hold off
xlabel('x'); ylabel('Probability Density');
```

step 2 查看图形。在输入程序代码后，按 "Enter" 键，得到的图形如图 3.19 所示。

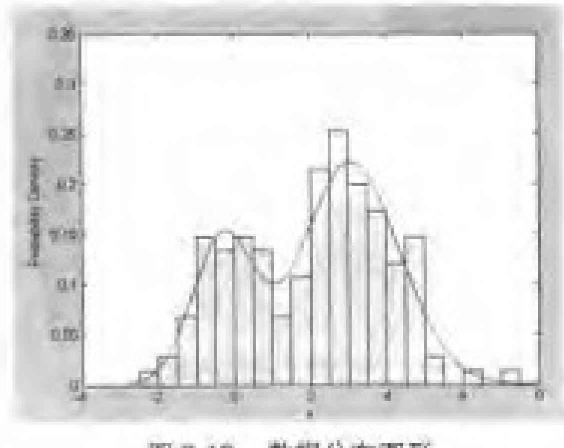


图 3.19 数据分布图形

3.7.4 假设检验

假设检验是数理统计中的一个重要内容，在 MATLAB 中可以实现多种常见概率分布的假设检验，例如单侧检验、双侧检验、均值检验、方差检验等多种。在本小节中，将以简单的例子来说明如何在 MATLAB 来进行假设检验。

例 3.39 对某试验数据进行平均值的正态分布单侧检验，总体的标准差已知，并且假设检验的置信水平为 5%，假设检验的平均值为 100。

step 1 进行假设检验。在 MATLAB 的命令窗口中输入下面的代码：

```
%设置假设检验的参数
mu0 = 100;
```

```
sig = 20;
N = 16;
%设置假设检验的置信水平
alpha = 0.05;
conf = 1-alpha;
%设置正态分布的截断点
cutoff = norminv(conf, mu0, sig/sqrt(N));
%产生数据点
x = [ linspace(90,cutoff), linspace(cutoff,127)];
y = normpdf(x,mu0,sig/sqrt(N));
%绘制正态分布图形
h1 = plot(x,y);
xhi = [ cutoff, x(x>=cutoff)];
yhi = [ 0, y(x>=cutoff)];
%绘制假设检验的面积图
patch(xhi,yhi,'b');
%设置图形的标题和坐标轴名称
title('Distribution of sample mean, N=16');
xlabel('Sample mean');
ylabel('Density');
text(96,.01,sprintf('Reject if mean>%.4g\nProb = 0.05',cutoff),'Color',
'b')
```

step 2 查看图形。在输入程序代码后，按"Enter"键，得到的图形如图 3.20 所示。

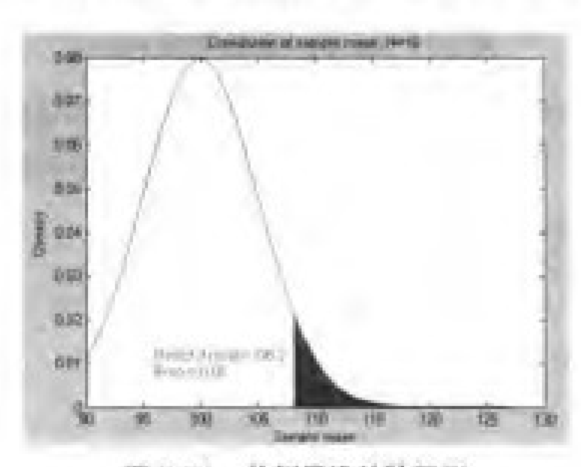


图 3.20 单侧假设检验图形



在上面的程序代码中，首先输入关于该假设检验的参数，然后根据正态分布反函数来求取满足假设检验条件的均值数值，最后在正态分布曲线中绘制出假设检验的图形。

step 3 修改假设检验的条件，进行假设检验。修改假设检验条件的均值为 110，重新进行假设检验。在 MATLAB 的命令窗口中输入下面的代码：

```
%修改假设检验的均值数值
mu1 = 110;
%计算正态分布数据点
y2 = normpdf(x,mu1,sig/sqrt(N));
%绘制正态分布图形
```

```
h2 = line(x,y2,'Color','r');
%绘制假设检验的面积图
yhi = [ 0, y2(x>=cutoff)];
patch(xhi,yhi,'r','FaceAlpha',0.25);
%添加图形的提示信息
P = 1 - normcdf(cutoff,mu1,sig/sqrt(N));
text(115,.06,sprintf('Reject if T>%.4g\nProb = %.2g',cutoff,P),'Color',
[1 0 0]);
legend([ h1 h2] ,'Null hypothesis','Alternative hypothesis');
```

step 4 查看图形。在输入程序代码后，按"Enter"键，得到的图形如图 3.21 所示。

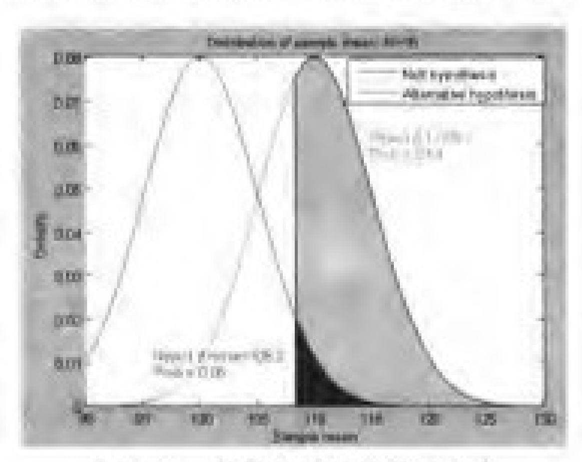


图 3.21　修改条件后的假设检验

step 5 绘制累积概率密度函数的图形。为了对比上面两种不同的假设检验条件，用户可以绘制概率密度函数的图形。在 MATLAB 的命令窗口中输入下面的程序代码

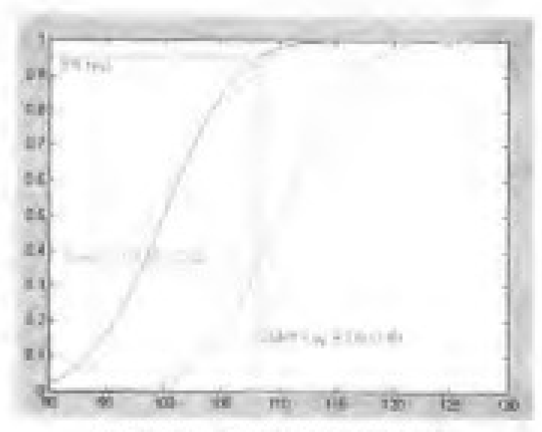
```
%计算累积概率密度函数
ynull = normcdf(x,mu0,sig/sqrt(N));
yalt = normcdf(x,mu1,sig/sqrt(N));

%绘制累积密度函数图形
plot(x,ynull,'b-',x,yalt,'r-');

%计算置信条件水平下的反正态分布数值
zval = norminv(conf);
cutoff = mu0 + zval * sig / sqrt(N);

%绘制图形
line([ 90,cutoff,cutoff] ,[ conf, conf, 0] ,'LineStyle',':');
msg = sprintf(' Cutoff = \\mu_0 + %.2g\\sigma / \\surd{n} ',zval);
text(cutoff,.15,msg,'Color','b');
text(min(x),conf,sprintf('    %g%% test',100*alpha),'Color','b',...
                          'verticalalignment','top')
palt = normcdf(cutoff,mu1,sig/sqrt(N));
line([ 90,cutoff] ,[ palt,palt] ,'Color','r','LineStyle',':');
text(91,palt+.02,sprintf(' Power is 1-%.2g = %.2g',palt,1-palt),'Color',
[ 1 0 0]);
```

step 6 查看图形。在输入程序代码后，按"Enter"键，得到的图形如图 3.22 所示。

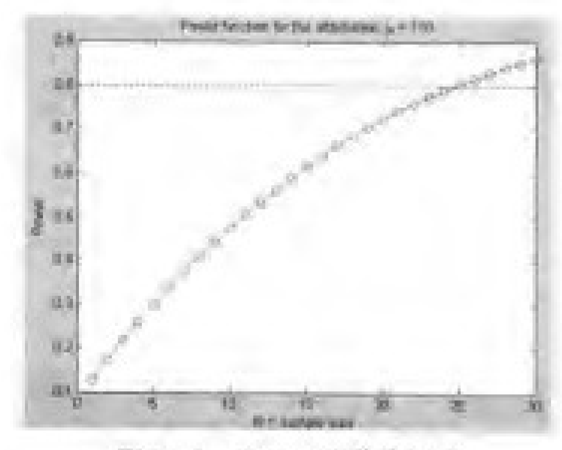

图 3.22 累积概率密度图形


上面的图形显示在两种不同的分布下的假设检验情况，实际上，可以在同一个图形中显示多个不同分布的假设检验情况。

step 7 绘制 Power 函数。Power 函数的定义是假设检验中拒绝检验的概率，绘制该函数的图形也可以查看不同的假设检验的情况。在 MATLAB 的命令窗口中输入下面的程序代码：

```
% 定义需要的 Power 参数数值
DesiredPower = 0.80;
Nvec = 1:30;
cutoff = mu0 + norminv(conf)*sig./sqrt(Nvec);
% 计算假设检验的 power 数值
power = 1 - normcdf(cutoff, mu1, sig./sqrt(Nvec));
% 绘制图形
plot(Nvec,power,'bo-',[0 30],[DesiredPower DesiredPower],'k:');
xlabel('N = sample size'); ylabel('Power')
title('Power function for the alternative: \mu = 110')
```

step 8 查看图形。在输入程序代码后，按 "Enter" 键，得到的图形如图 3.23 所示。


图 3.23 Power 函数的图形

step 9 使用 Monte Carlo 模拟来检验 Power 函数的检验结果。在 MATLAB 的命令窗口中输入下面的程序代码：

```
% 定义 Monte Carlo 模拟的参数
```

```
%nsamples 表示样本数值
%smaplenum 是样本中数据点

nsamples = 400;
samplenum = 1:nsamples;
N=25;

% 创建零值矩阵
h0 = zeros(1,nsamples);
h1 = h0;

% 进行右侧已知方差条件下均值假设检验
for j=1:nsamples
    Z0 = normrnd(mu0,sig,N,1);
    h0(j) = ztest(Z0,mu0,sig,alpha,'right');
    Z1 = normrnd(mu1,sig,N,1);
    h1(j) = ztest(Z1,mu0,sig,alpha,'right');
end
p0 = cumsum(h0) ./ samplenum;
p1 = cumsum(h1) ./ samplenum;
% 绘制对应的图形
plot(samplenum,p0,'b-',samplenum,p1,'r-')
xlabel('Sample number'); ylabel('Proportion significant')
title('Verification of power computation')
legend('Null hypothesis','Alternative hypothesis','Location','East')
```

step 10 查看图形。在输入程序代码后，按"Enter"键，得到的图形如图 3.24 所示。

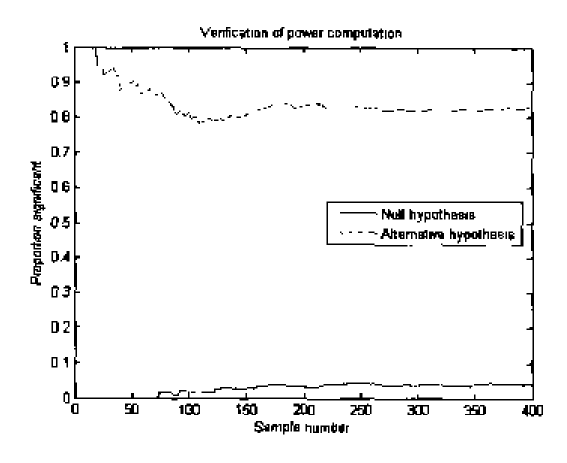


图 3.24 Monte carlo 检验图形

小结

在本章中，依次向读者介绍了矩阵分析、矩阵分解、特征值计算、数值积分、数理统计等内容，这些内容是 MATLAB 进行数值运算的重要部分，其中关于矩阵的分析和运算是其他操作的基础内容，希望用户能够熟练掌握。下一章，将介绍如何使用 MATLAB 进行数据分析。

◆ ◆ ◆

第4章 数据分析

本章包括

- 数据插值
- 曲线拟合
- 傅里叶（Fourier）分析
- 优化
- 常微分方程

数据分析和处理在各个领域有着广泛的应用，尤其是在数学、物理等科学领域和工程领域的实际应用中，会经常遇到进行数据分析的情况。例如，在工程领域根据有限的已知数据对未知数据进行推测时经常需要用到数据插值和拟合的方法，在信号工程领域则经常需要用到傅里叶变换的工具，在物理或工程领域中则经常需要根据现实情况抽象出微分方程，进行数值求解，等等。这些常见的数据分析方法在MATLAB中都可以实现，同时，MATLAB本身有着强大的数据分析和处理功能，还可以处理类似优化、偏微分方程等比较复杂的问题。

对于熟悉MATLAB基本命令的读者而言，通过本章的学习，可以了解到各命令的使用场合以及内在关系，同时获得综合运用命令解决实际问题的思路和借鉴的经验。本章将分别介绍插值、拟合、傅里叶变换、常微分方程、优化和偏微分方程等各种内容，从总体上来讲，本章各节之间没有依从关系，读者可以根据需要选择阅读相关章节，同时，本章中所列举的各个例子都是独立完整的，用户可以在自己的机器上运行。

插值

插值（Interpolation）是指在所给的基准数据情况下，研究如何平滑地估算出基准数据之间其他点的函数数值。每当其他点上函数值若获取的代价比较高时，插值就会发挥作用。

在MATLAB中提供多种插值函数，这些插值函数在获得数据的平滑度、时间复杂度和空间复杂度方面有着完全不同的性能。本节将主要介绍一维插值命令interp1，二维插值命令interp2等MATLAB内置函数。同时，还将根据不同的插值方法自行编写M文件，完成不同的插值运算，例如Lagrange插值、Newton插值等，下面分小节详细介绍。



一维插值命令

在MATLAB中，一维插值表示的是对一维函数$y=f(x)$进行插值。为了让读者更加形象地了解一维插值的含义，下面列出一维插值的图形，如图4.1所示。

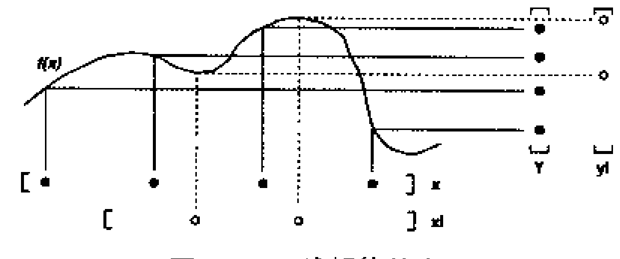


图4.1 一维插值的含义

在上面的图形中，实心点(x, y)表示已知数据点，空心点(x_i, y_i)中的横坐标x_i代表的是需要估计数值的位置，纵坐标y_i表示插值后运算的数值。

在 MATLAB 中，一维多项式插值的方法通过命令 interp1 实现，其具体的调用格式如下：

- yi = interp1(x,Y,xi)　在该命令中，x 必须是向量，y 可以是向量也可以是矩阵。如果 y 是向量，则必须与变量 x 具有相同的长度，这时 xi 可以是标量，也可以是向量或者矩阵；
- yi = interp1(Y,xi)　在默认情况下，x 变量选择为 1:n，其中 n 是向量 y 的长度；
- yi = interp1(x,Y,xi,method)　输入变量 method 用于指定插值的方法，具体的插值方法将在本章后面的内容中详细介绍；
- yi = interp1(x,Y,xi,method,'extrap')　对超出数据范围的插值运算使用外推方法；
- yi = interp1(x,Y,xi,method,extrapval)　对超出数据范围的插值数据返回 extrapval 数值，一般为 NaN 或者 0；
- pp = interp1(x,Y,method,'pp')　返回数值 pp 为数据 y 的分段多项式形式、method 指定产生分段多项式 pp 的方法。



在 MATLAB 中，还提供 interp1q 命令作为 interp1 的快速命令。对于数据间隔不均的数据，interp1q 命令运行的速度比 interp1 命令要快，主要原因在于 interp1q 命令没有对输入数据进行检测。关于该命令的详细使用方法请参看相应的帮助文件。

在上面的命令中，method 参数的取值和对应的含义如下：

- **nearest**：最邻近插值方法（Nearest neighbor interpolation），这种插值方法在已知数据的最邻近点设置插值点，对插值点的数值进行四舍五入，对超出范围的数据点返回 NaN；
- **linear**：线性插值（Linear interpolation），这是 interp1 命令中 method 的默认数值。该方法采用直线将相邻的数据点相连，对超出数据范围的数据点返回 NaN；
- **spline**：三次样条插值（Cubic spline interpolation），该方法采用三次样条函数获取插值数据点，在已知点为端点的情况下，插值函数至少具有相同的一阶和二阶导数；
- **pchip**：分段三次厄米多项式插值（Piecewise cubic Hermite interpolation）；
- **cubic**：三次多项式插值，与分段三次厄米多项式插值方法相同；
- **v5cubic**：MATLAB5 中使用的三次多项式插值。

4.1.2 一维插值实例

为了让读者直观地了解上面各个不同插值方法的差别，下面举例演示不同方法的插值。

例 4.1　使用不同的方法来对基础数据进行插值运算，并比较各种方法的不同。

step 1 某城市在 1900－1990 年中，每隔 10 年统计该城市的人口数量，得到的结果如下：

```
75.995、91.972、105.711、123.203 、131.669；
150.697、179.323、203.212、226.505、249.633
```

上面数据的单位是百万。需要以上面的数据为基础，对没有进行人口统计的年限的人口数量进行推测，分别使用不同的插值方法。

step 2 选择命令窗口编辑栏中的"File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
%已知数据
t = 1900:10:1990;
p = [ 75.995  91.972  105.711  123.203  131.669...
     150.697  179.323  203.212  226.505  249.633];
%使用不同的方法进行插值运算
x = 1900:0.01:1990;
yi_linear=interp1(t,p,x);
yi_spline=interp1(t,p,x,'spline');
yi_cubic=interp1(t,p,x,'cubic');
yi_v5cubic=interp1(t,p,x,'v5cubic');
%绘制对比图形
subplot(2,1,1);
plot(t,p,'ko');hold on;
plot(x,yi_linear,'g','LineWidth',1.5);hold on
plot(x,yi_spline,'y','LineWidth',1.5);
grid on;
title('Linear Vs Spline');
subplot(2,1,2);
plot(t,p,'ko');hold on;
plot(x,yi_cubic,'m','LineWidth',1.5);hold on
plot(x,yi_v5cubic,'k','LineWidth',1);
grid on;
title('Cubic Vs V5Cubic');
%创建新的图形窗口
figure
yi_nearest=interp1(t,p,x,'nearest');
plot(t,p,'ko');hold on;
plot(x,yi_nearest,'g','LineWidth',1.5);hold on
grid on;
title('Nearest Method')
```

当输入上面的代码后，将该代码保存为"intexa.m"文件。

step 3 查看计算结果。在命令窗口中输入"intexa"，然后按"Enter"键，得到的对比结果如图 4.2 所示。同时，使用"Nearest"插值方法得到的结果如图 4.3 所示。

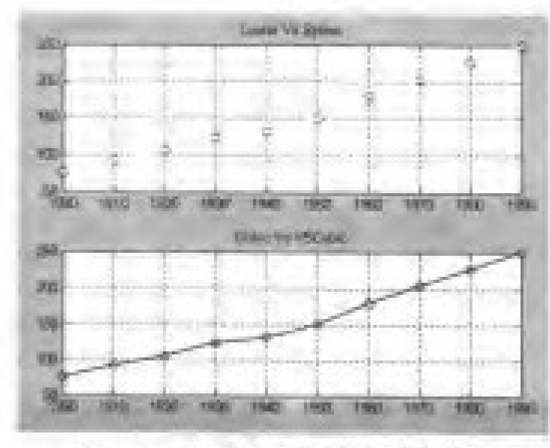


图 4.2 不同插值方法得到的结果

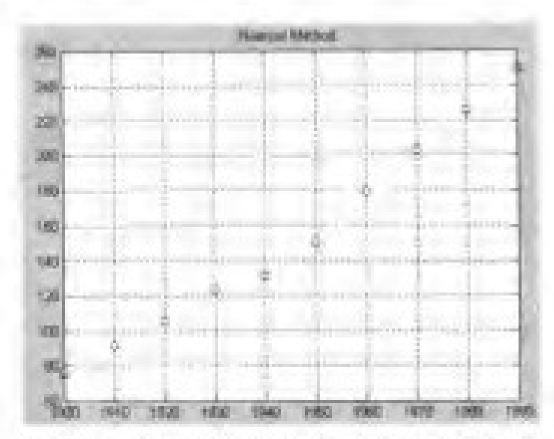


图 4.3 使用 "Nearest" 方法得到的结果

step 4 根据不同的插值方法，估计中间年限的人口数目。选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
msg='       year      Cubic        Linear        Nearest        Spline';
for i=0:8
n=10*i;
year=1905+n;
pop(i+1,1)=year;
pop(i+1,2)=yi_cubic((year-1900)/0.01+1);
pop(i+1,3)=yi_linear((year-1900)/0.01+1);
pop(i+1,4)=yi_nearest((year-1900)/0.01+1);
pop(i+1,5)=yi_spline((year-1900)/0.01+1);
end
P=round(pop);

disp(msg)
disp(P)
```

在输入上面的程序代码后，将该代码保存为 "runexa.m" 文件。

step 5 查看计算结果。在命令窗口中输入 "runexa"，然后按 "Enter" 键，得到的结果如下：

```
     year      Cubic        Linear        Nearest        Spline
     1905          84            84            92            85
     1915          99            99           106            98
     1925         115           114           123           115
     1935         127           127           132           128
     1945         140           141           151           139
     1955         165           165           179           165
     1965         192           191           203           192
     1975         215           215           227           215
     1985         238           238           250           238
     1985         238           238           250           238
```



从上面的结果可以看出，使用不同的插值方法得到的预测数据会略有差别。上面的结果中，"Nearest" 方法得到的结果是偏差最大的一种，其他方法则差别不大。

根据上面的实例，下面简要比较各种插值在执行速度、占用内存大小以及获得数据的平滑度方面的性能，如表 4.1 所示。

表 4.1 各种插值方法的比较

方法	说明
Nearest	最快的插值方法，但是数据平滑方面最差，数据是不连续的
Linear	执行速度较快，有足够的精度。最为常用，而且为默认设置
Cubic	较慢，精度高，平滑度好，当希望得到平滑的曲线时可以使用该选项
Spline	执行速度最慢，精度高，最平滑



在上面小节中，多次使用到了MATLAB中关于M文件中的基础知识来实现各种插值方法的功能，关于M文件的使用方法请参看相应的章节。

4.1.3 二维插值命令

二维插值是高维插值的一种，主要应用于图像处理和数据的可视化，其基本思想与一维插值大致相同，它是对两个变量的函数 $z=f(x, y)$进行插值。其对应的插值原理示意如图 4.4 所示。

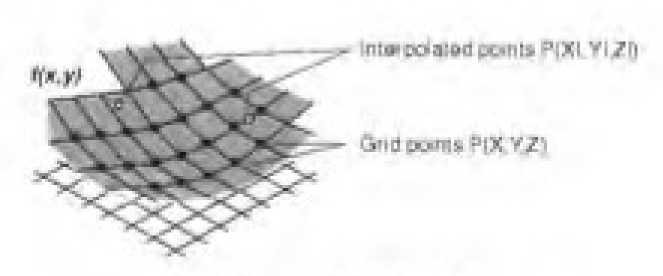


图 4.4 二维插值示意图

在 MATLAB 中，二维插值的常用函数是 interp2，其调用格式如下：

- ZI = interp2(X,Y,Z,XI,YI) 原始数据 x、y 和 z 决定插值函数 $z=f(x, y)$，返回的数值 **ZI** 是(**XI**, **YI**)在函数 $z=f(x, y)$上的数值；
- ZI = interp2(Z,XI,YI) 其中如果 Z 的维度是 $n \times m$，则有 $x=1:n$，$y=1:m$；
- ZI = interp2(Z,ntimes) 在两点之间递归地进行插值 ntimes 次；
- ZI = interp2(X,Y,Z,XI,YI,method) 使用选定的插值方法进行二维插值。

关于上面命令中的参数，笔者认为有必要提示读者下面的情况。

- 对于其他高维插值的命令，使用格式与此大致相同。
- 在上面的命令中，X、Y和Z是进行插值的基准数据。XI、YI是待求插补函数值ZI的自变量对组，虽然随着维度的增加，插值的具体操作变得越来越复杂，但是基本原理和一维插值的情况相同，调用名称也很类似。
- 参数X和Y必须满足一定的条件，首先X和Y必须是同维矩阵，同时矩阵中的元素，无论行向还是列向，都必须单调排列。最后，X和Y必须是Plaid格式的矩阵，所谓Plaid格式的矩阵，是指X矩阵每一行的元素依照单调次序排列，但是任何两行都是相同的；Y矩阵每一列的元素依照单调次序排列，但是任何两列都是相同的。
- 对于XI和YI数据系列，一般需要使用 meshgrid 命令来创建。具体的方法为：给定两个变量的分度向量，然后通过 meshgrid 命令产生对应的数据系列。



关于该命令的 method 参数，其含义和前面小节中 interp1 命令中的 method 参数类似，只是参数数值 linear 表示的是双线性插值方法。

4.1.4 二维插值实例

例 4.2 以二元函数 $z(x,y)=\dfrac{\sin\sqrt{x^2+y^2}}{\sqrt{x^2+y^2}}$ 为例，首先经过"仿真"获取基础数据，生成一组基础的稀疏"测量数据"，然后使用二维插值得到更细致的数据，完成绘图。

step 1 选择命令窗口编辑栏中的"File" ⇨ "New" ⇨ "M-File"命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
%生成基础测量数据
x=-3*pi:3*pi;
y=x;
[X,Y]=meshgrid(x,y);
R=sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
[dzdx,dzdy]=gradient(Z);
dzdr=sqrt(dzdx.^2+dzdy.^2);
%绘制基础数据图形
surf(X,Y,Z,abs(dzdr))
colormap(spring)
alphamap('rampup')
colorbar
```

完成上面的程序代码后，将程序代码保存为"intexp3.m"文件。

step 2 查看程序结果。在命令窗口中输入"intexp3"后，按"Enter"键，得到的结果如图 4.5 所示。

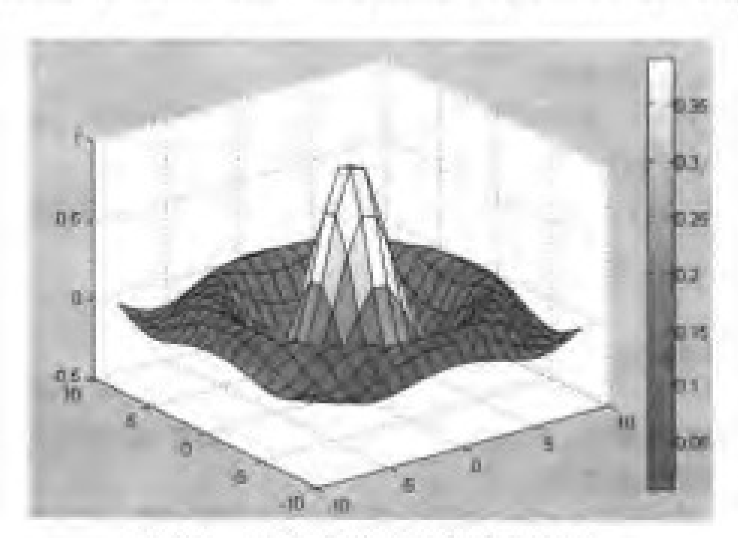


图 4.5 根据基础数据绘制的三维图形

step 3 添加二维插值的命令代码。打开上面步骤保存的 intexp3 文件，然后在 M 文件编辑器中添加下面的程序代码：

```
%进行二维插值运算
xi=linspace(-3*pi,3*pi,100);
yi=linspace(-3*pi,3*pi,100);
```

```
[XI,YI]=meshgrid(xi,yi);
ZI=interp2(X,Y,Z,XI,YI,'cubic');
%绘制插值后的数据图形
figure
surf(XI,YI,ZI)
colormap(spring)
alphamap('rampup')
colorbar
```

输入上面的命令后，保存程序代码，完成完整的"intexp3.m"文件。

step 4 查看程序代码结果。在命令窗口中输入"intexp3"后，按"Enter"键，得到的结果如图 4.6 所示。

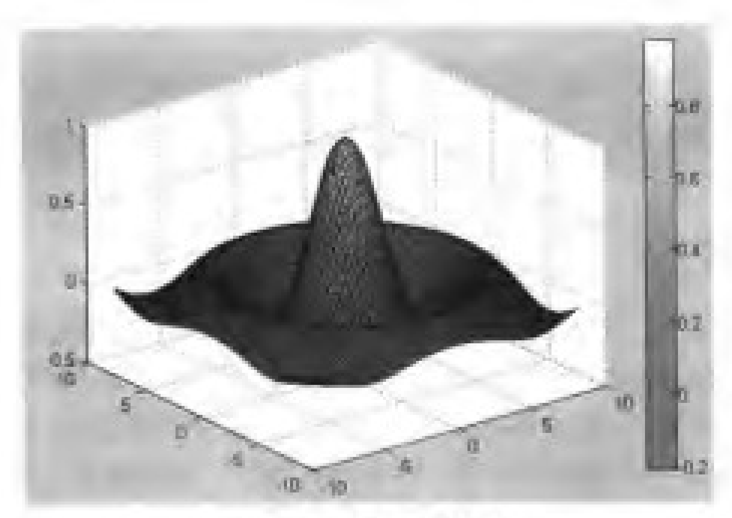


图 4.6 完成插值后的图形

例 4.3 在三维图形表面添加等高线，使用二维插值的方法计算等高线的数据点坐标，同时使用 Streamslice 命令绘制三维图形表面等高线数据点的梯度。

step 1 在 MATLAB 的命令窗口中输入下面的代码：

```
%创建基础数据
>> z = peaks;
>>surf(z)
shading interp
hold on
%添加三维图形表面的等高线
[c ch] = contour3(z,20); set(ch,'edgecolor','b')
%计算三维图形表面的数值梯度
[u v] = gradient(z);
h = streamslice(-u,-v);
set(h,'color','k')
%对等高线进行二维数值插值
for i=1:length(h);
    zi = interp2(z,get(h(i),'xdata'),get(h(i),'ydata'));
    set(h(i),'zdata',zi);
end
view(30,50); axis tight
```

step 2 查看程序结果。输入程序代码后，按"Enter"键，得到的结果如图 4.7 所示。

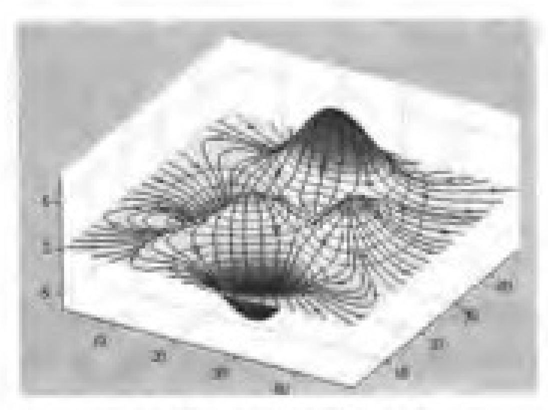


图 4.7　完成的三维图形



说明　在上面的两个例子中，多次用到关于MATLAB图形对象操作的命令，主要涉及图形颜色、角度等属性，关于这些命令的详细使用方法读者可以查看相关的章节。

4.1.5　样条插值

除了前面提到的一维和二维插值方法之外，MATLAB还提供样条插值的方法。样条函数的主要思想是，假设有一组已知的数据点，目标是找到一组拟合多项式，在多项式拟合的过程中，对于每组相邻的样本数据，用三次多项式去拟合样本数据点之间的曲线。为了保证拟合的唯一性，对这个三次多项式用在样点处的一阶、二阶导数加以约束。因此，除了研究区间的端点之外，所有样本点之间的数据也能保证连续的一阶和二阶导数。

关于样条插值的主要命令如下：

- yy = spline(x,y,xx)
- pp = spline(x,y)
- yy = ppval(pp,xx)

在上面的命令中，(x,y)表示的是样点数据，xx则是插值的数据点系列。下面将利用几个简单的例子来说明 Spline 函数的使用方法。

例4.4　对基础数据进行样条插值运算，分别使用样条函数和一维插值命令，并对插值结果进行比较。

step 1　在 MATLAB 的命令窗口中输入下面的代码：

```
>>x = -4:4;
>>y = [ 0 .15 1.12 2.36 2.36 1.46 .49 .06 0];
>>cs = spline(x,[ 0 y 0] );
>>xx = linspace(-4,4,101);
>>yy= ppval(cs,xx);
>>yyt=interp1(x,y,xx,'spline');
>>plot(x,y,'o',xx,yy,'g.');hold on
>>plot(xx,yyt,'m','LineWidth',1.5)
>>grid on
```

step 2　查看程序代码的结果。当输入上面的代码后，按 "Enter" 键，得到的结果如图 4.8 所示。

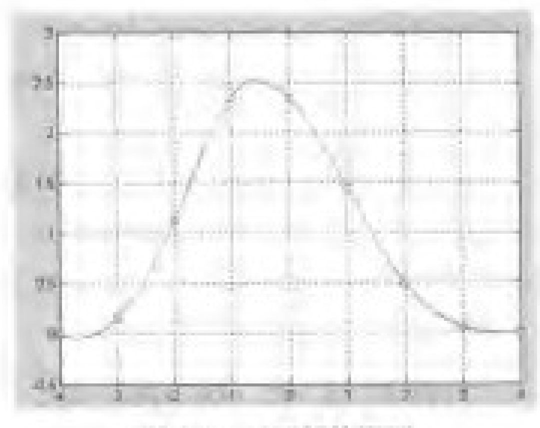


图 4.8 显示插值图形

例 4.5 使用样条插值进行圆形数据插值。

step 1 在 MATLAB 的命令窗口中输入下面的代码：

```
>> x = pi*[ 0:.5:2];
>>y =[0  1  0 -1  0  1  0;
      1  0  1  0 -1  0  1];
>>pp = spline(x,y);
>>yy = ppval(pp, linspace(0,2*pi,101));
>>plot(yy(1,:),yy(2,:),'-b',y(1,2:5),y(2,2:5),'or');
>>grid on
>> axis equal
```

step 2 查看程序代码的结果。当输入上面的代码后，按"Enter"键，得到的结果如图 4.9 所示。

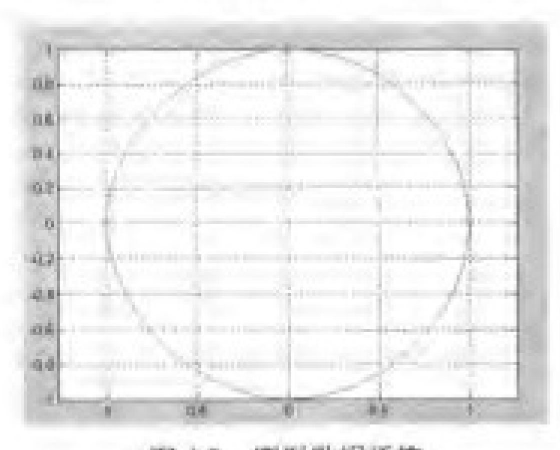


图 4.9 圆形数据插值

4.1.6 牛顿插值原理

插值一直是工程和科学中的重要内容，根据工程中不同的实践要求，已经发展出多种插值运算方法。因此，除了 MATLAB 内置的插值函数之外，还可以根据实践需要编写插值运算的 M 文件，然后将其运算在其他合适的领域。在本小节中，将详细介绍如何使用牛顿（Newton）插值方法进行数据插值。

首先，读者有必要了解牛顿（Newton）插值方法的基本原理。对于 N+1 个已知数据系列 $\{(x_0, y_0),(x_1, y_1), \cdots,(x_N, Y_N)\}$，可以被下面的 N−1 阶的多项式循环引用，满足下面的 N 个数据系列 $\{(x_0, y_0),(x_1, y_1), \cdots,(x_{N-1}, Y_{N-1})\}$的等式：

$$n_N(x)=a_0+a_1(x-x_0)+a_2(x-x_0)(x-x_1)+\cdots$$
$$=n_{N-1}(x)+a_N(x-x_0)(x-x_1)\cdots(x-x_{N-1})$$

同时，需要满足的约束条件为 $n_0(x)=a_0$。

根据上面的等式方程组，得到的牛顿插值的数值系数满足下面的等式：

$$a_0=y_0$$
$$a_1=\frac{y_1-a_0}{x_1-x_0}=\frac{y_1-y_0}{x_1-x_0}=Df_0$$
$$a_2=\frac{\frac{y_2-y_1}{x_2-x_1}-\frac{y_1-y_0}{x_1-x_0}}{x_2-x_0}=\frac{Df_1-Df_0}{x_2-x_0}=D^2f_0$$

根据数学归纳法，得到牛顿插值方法的系数通式为

$$a_N=\frac{D^{N-1}f_1-D^{N-1}f_0}{x_N-x_0}=D^Nf_0$$



关于上面系数运算的过程，需要使用到一些基础的数学知识，限于篇幅，分析过程就不在这里详细介绍了。

4.1.7 牛顿插值实例

本小节将使用一个简单的实例，说明如何使用牛顿插值法进行数值插值。

例4.6 以函数 $f(x)=\frac{1}{1+8x^2}$ 为基准函数，分别采用不同阶数的牛顿插值多项式，比较不同的阶数的插值误差。

step 1 选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开M文件编辑器，在其中输入下面的程序代码：

```
function [n,DD]=newtonp(x,y)
% Newton 插值函数
% 输入参数  x=[x0 x1 ... xN]
%           y=[y0 y1 ... yN]
% 输出参数  n=Newton 系数

N=length(x)-1;
DD=zeros(N+1,N+1);
DD(1:N+1,1)=y';
for k=2:N+1
    for m=1:N+2-k
        DD(m,k)=(DD(m+1,k-1)-DD(m,k-1))/(x(m+k-1)-x(m));
    end
end
a=DD(1,:);
n=a(N+1);
for k=N:-1:1
    n=[n a(k)]-[0 n*x(k)];
end
```

将上面的程序代码保存为 "newtonp.m" 文件，在后面的步骤中需要引用上面的程序代码进行牛顿插值运算。

step 2 在 MATLAB 的命令窗口中输入下面的代码：

```
%计算 4 阶牛顿方法
x=[ -1 -0.5 0 0.5 1];
y=1./(1+8*x.^2);
n=newtonp(x,y);
xx=[ -1:0.02:1];
yy1=polyval(n,xx);
%计算原始函数的数据
yy=1./(1+8*xx.^2);
%绘制图形
plot(xx,yy,'k-',x,y,'o');
hold on;
%绘制插值图形
plot(xx,yy1,'b','LineWidth',1.5);
hold on;
%计算 8 阶牛顿方法
x1=[ -1:0.25:1];
y1=1./(1+8*x1.^2);
n1=newtonp(x1,y1);
yy2=polyval(n1,xx);
%绘制新的插值图形
plot(x1,y1,'kd');
hold on;
plot(xx,yy2,'m','LineWidth',1.5);
hold on;
%计算 10 阶牛顿方法
x2=[ -1:0.2:1];
y2=1./(1+8*x2.^2);
n2=newtonp(x2,y2);
yy3=polyval(n2,xx);
plot(x1,y1,'rh');
hold on;
plot(xx,yy3,'g','LineWidth',1.5);
grid on;
%添加图形的标题
title('Newton Interpolation')
hold off
```

step 3 查看程序代码的结果。在输入上面的代码后，按"Enter"键，得到牛顿插值的结果，如图 4.10 所示。

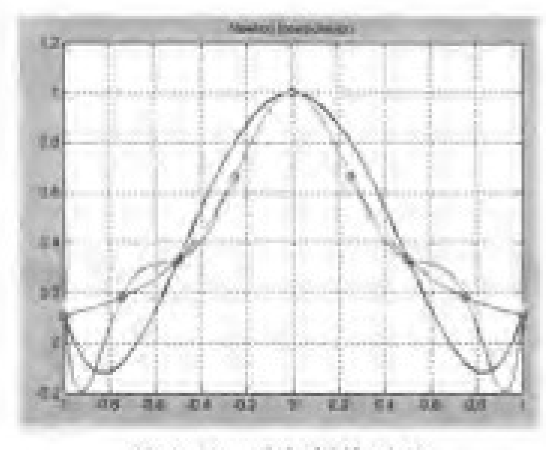


图 4.10　牛顿插值图形

step 4 进行插值误差分析。在 MATLAB 的命令窗口中输入下面的代码：

```
figure
plot(xx,yy1-yy,'b','LineWidth',1.5);hold on;
plot(xx,yy2-yy,'m','LineWidth',1.5);hold on;
plot(xx,yy3-yy,'g','LineWidth',1.5);
grid on
title('The error of Newton Interpolation')
```

step 5 查看程序代码的结果。当输入上面的代码后，按"Enter"键，得到的结果如图 4.11 所示。

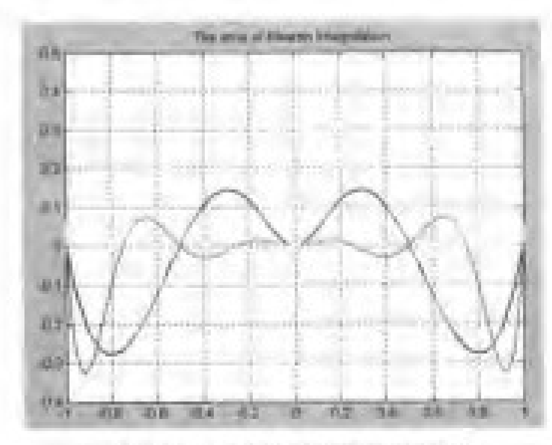


图 4.11　牛顿插值的误差结果



从上节的结果中可以看出，增加牛顿插值多项式的阶数，并不能显著地减少插值的误差。同时，在插值数值范围两侧出现明显的振荡现象。这种现象成为插值情况的 Runge 现象。因此，牛顿插值在 5 阶以上就很少使用了。

4.1.8 Chebyshev 多项式插值原理

在前面所介绍的插值方法中，主要研究的是如何在已知数据条件下，如何在各种约束条件下完成不同的插值结果。在插值问题中，还涉及另外一个层次的问题，也就是如何选择插值所需要的基础数据点。前面例子中，选择的基础数据点都是等分数据点，但是，这种方法并不是最佳的。在本小节中，还是以上面小节的实例为例子，也就是以函数 $f(x)=\frac{1}{1+8x^2}$ 为基准函数，选择不同间距的基准数据，然后使用基础数据进行整体数据的插值工作。

根据上面小节中的例子，由于在数据范围两侧出现了振荡现象，因此，如果在此例中在两侧选择数据点的密度大于在中间范围选择数据点，这样就可以很有效地提高数据插值的精度。Chebyshev 多项式提供下面的数据选择方法。

在标准化的数据范围[-1, +1] 内选择下面的数据点：

$$x_k=\cos\frac{2N+1-2k}{2(N+1)}\pi$$

在上面的数据系列中，参数 k=0, 1, …, N。对于任意一个有限的区间[a, b]内，存在下面的等式关系：

$$x_k=\frac{b-a}{2}x_k+\frac{a+b}{2}=\frac{b-a}{2}\cos\frac{2N+1-2k}{2(N+1)}\pi+\frac{a+b}{2}$$

满足上面关系式的基础数据点系列满足下面的图形，如图 4.12 所示。

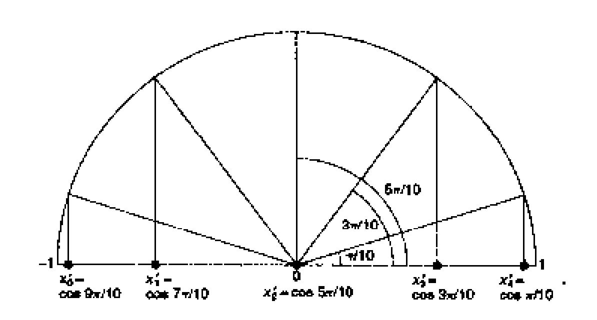


图4.12 选择基础数据点



Chebyshev 多项式插值实例

例4.7 以函数$f(x)=\dfrac{1}{1+8x^2}$为基准函数，使用上面介绍的选择数据点的方法选择基础数据点，然后进行牛顿插值，最后比较不同的阶数的插值误差。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
%计算4阶插值分析
N=4;
k=[ 0:N] ;
%选择基础数据点系列
x=cos((2*N+1-2*k)*pi/2/(N+1));
y=1./(1+8*x.^2);
c=newtonp(x,y);
xx=[ -1:0.02:1] ;
yy=1./(1+8*xx.^2);
yy1=polyval(c,xx);
plot(xx,yy,'k-',x,y,'o');
hold on;
plot(xx,yy1,'b','LineWidth',1.5);
%计算8阶插值分析
N=8;
k=[ 0:N] ;
x=cos((2*N+1-2*k)*pi/2/(N+1));
y=1./(1+8*x.^2);
c1=newtonp(x,y);
plot(x,y,'*');
hold on;
plot(xx,yy2,'m','LineWidth',1.5);
%计算10阶插值分析
N=10;
k=[ 0:N] ;
x=cos((2*N+1-2*k)*pi/2/(N+1));
y=1./(1+8*x.^2);
c3=newtonp(x,y);
yy3=polyval(c3,xx);
plot(x,y,'d');
hold on;
plot(xx,yy3,'g','LineWidth',1.5);
grid on
Title('Chebyshev Interpolation')
```

step 2 查看程序代码的结果。当输入上面的代码后，按 "Enter" 键，得到的结果如图 4.13 所示。

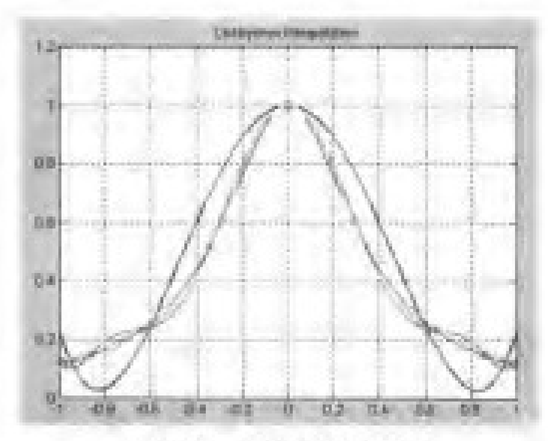


图 4.13　完成的插值结果

step 3 在 MATLAB 的命令窗口中输入下面的程序代码：

```
figure;
plot(xx,yy1-yy,'b','LineWidth',1.5);hold on;
plot(xx,yy2-yy,'m','LineWidth',1.5);hold on;
plot(xx,yy3-yy,'g','LineWidth',1.5);
grid on
title('The error of Chebyshev Interpolation')
legend('4阶','8阶','10阶')
```

step 4 查看程序代码的结果。当输入上面的代码后，按 "Enter" 键，得到的结果如图 4.14 所示。

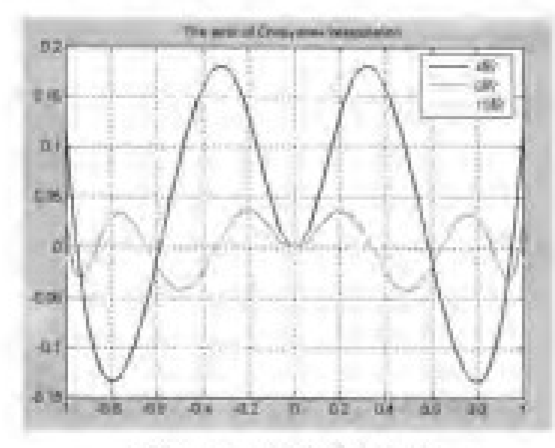


图 4.14　各阶插值误差



根据上面的实例结果，当使用 Chebyshev 的方法来选择基础数据点时，可以十分有效地消除 Runge 的现象，提高数值插值的效率。

4.2　曲线拟合

在科学和工程领域，曲线拟合的主要功能是寻求平滑的曲线来最好地表现带有噪声的测量数据，从这些测量数据中寻求两个函数变量之间的关系或者变化趋势，最后得到曲线拟合的函数表达

式 $y=f(x)$。从前面的“插值”小节中可以看出，使用多项式进行数据拟合会出现数据振荡或者Runge的现象，而Spline插值的方法可以得到很好的平滑效果，但是关于该插值方法有太多的参数，不适合曲线拟合的方法。

同时，由于在进行曲线拟合的时候，已经认为所有测量数据中已经包含噪声，因此，最后的拟合曲线并不要求通过每一个已知数据点，衡量拟合数据的标准则是整体数据拟合的误差最小。一般情况下，MATLAB的曲线拟合方法使用的是“最小方差”函数，其中方差的数值是拟合曲线和已知数据之间的垂直距离。和插值相同，对于曲线拟合也存在着各种不同的标准。为了满足不同的拟合要求，用户同样需要自行编写对应的M文件。

本节将分别介绍几种比较常见的曲线拟合方法，最后，还将介绍MATLAB提供的曲线拟合界面操作方法。

4.2.1 多项式拟合

在MATLAB中，提供polyfit函数来计算多项式拟合系数，其设定曲线拟合的目标是最小方差（Least Squares）或者被称为最小二乘法。Polyfit函数的调用格式如下：

- [p,S,mu] = polyfit(x,y,n)
- [y,delta] = polyval(p,x,S,mu)

其中，x和y表示的是已知测量数据，n是多项式拟合的阶数。同时，参数μ满足等式 $\hat{x}=\frac{x-\mu_1}{\mu_2}$，其中 $\mu_1=\text{mean}(x)$，$\mu_2=\text{std}(x)$ 而且 $\mu=[\mu_1,\mu_2]$。

通过上面的命令，最后可以得到的拟合曲线的多项式为：

$$y=p_1x^n+p_2x^{n-1}+\cdots+p_nx+p_{n+1}$$



由于最小方差的标准是比较常见的拟合要求，这里就不详细展开了，感兴趣的读者可以查看对应的数学书籍，或者查看polyfit的M文件中的程序代码。

例 4.8　使用polyfit命令来进行多项式的数值拟合，并分析曲线拟合的误差情况。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>> x = (0: 0.1: 5)';
>>y = erf(x);
% 计算多项式拟合的参数
>> [p,s] = polyfit(x,y,6);
>> [yp,delta] = polyval(p,x,s);
>>plot(x,y,'+',x,yp,'g-',x,yp+2*delta,'r:',...
x,yp-2*delta,'r:'), grid on
>> axis([0  5  0  1.4]);
>> Title('Polynomial curve fitting')
>> legend('Original','Fitting')
```

step 2 查看程序代码的结果。在输入上面的代码后，按“Enter”键，得到的曲线拟合图形如图4.15所示。

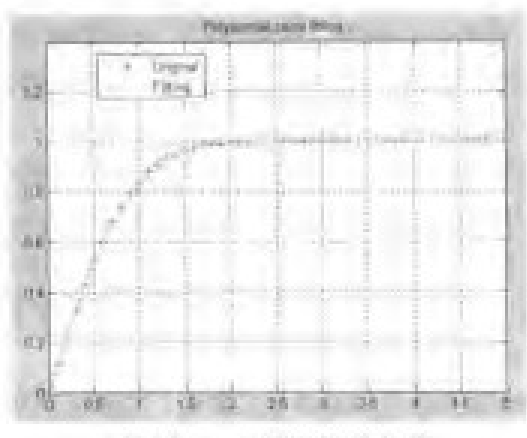


图 4.15　多项式拟合结果

上面的例子中使用了函数erf来产生基础数据。该函数是MATLAB的内置误差函数，其表达式为 $\mathrm{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\mathrm{d}t$。关于该函数的详细知识，可以查看对应的帮助文件。



上面的例子并不复杂，如果能够了解上面例子的各个参数的含义，就能加深对polyfit命令各参数的含义。

4.2.2 加权最小方差（WLS）拟合原理

所谓加权最小方差（Weighted Least Squares），就是根据基础数据本身各自的准确度（或者被称为可靠性）的不同，在拟合的时候给每个数据以不同的加权数值。这种方法比前面所介绍的单纯最小方差方法要更加符合拟合的初衷。

对于N阶多项式的拟合公式，所需要求解的拟合系数需要求解线性方程组，其中线性方程组的系数矩阵和需要求解的拟合系数（也就是变量）矩阵分别为：

$$A=\begin{pmatrix} x_1^N & \cdots & x_1 & 1 \\ x_2^N & \cdots & x_2 & 1 \\ \vdots & \cdots & \vdots & \vdots \\ x_M^N & \cdots & x_M & 1 \end{pmatrix},\quad \theta=\begin{pmatrix} \theta_N \\ \theta_{N-1} \\ \vdots \\ \theta_1 \end{pmatrix}$$

使用加权最小方差（WLS）方法求解得到的拟合系数为：

$$\theta_W^c=\begin{pmatrix} \theta_{WN}^c \\ \theta_{WN-1}^c \\ \vdots \\ \theta_1^c \end{pmatrix}=[A^TWA]^{-1}A^TWy$$

其对应的加权最小方差为表达式 $J_W=[A\theta-y]^TW[A\theta-y]$ 最小。



关于上面的拟合系数的详细求解过程，由于涉及比较高级的线性代数的知识，在这里限于篇幅就不详细展开分析了，感兴趣的读者请查看相应的书籍。

4.2.3 加权最小方差（WLS）拟合实例

例 4.9　根据上面的WLS数据拟合方法，自行编写使用WLS方法拟合数据的M函数，然后使用

WLS方法进行数据拟合。

step 1 选择命令窗口编辑栏中的“File” ⇨ "New” ⇨ “M-File”命令，打开M文件编辑器，在其中输入下面的程序代码：

```
function [ th,err,yi] =polyfits(x,y,N,xi,r)
%x,y  :数据点系列
%N    :多项式拟合的系统
%r    :加权系数的逆矩阵

M=length(x);
x=x(:);
y=y(:);

%判断调用函数的格式
if nargin==4
%当用户调用函数格式为(x,y,N,r)
    if length(xi)==M
        r=xi;
        xi=x;
%当用户调用函数格式为(x,y,N, xi)
    else r=1;
    end
%当用户调用函数格式为(x,y,N)
elseif nargin==3
    xi=x;
    r=1;
end
%求解系数矩阵
A(:,N+1)=ones(M,1);
for n=N:-1:1
    A(:,n)=A(:,n+1).*x;
end

if length(r)==M
    for m=1:M
        A(m,:)=A(m,:)/r(m);
        y(m)=y(m)/r(m);
    end
end

%计算拟合系数
th=(A\y)';
ye=polyval(th,x);
err=norm(y-ye)/norm(y);
yi=polyval(th,xi);
```

在输入上面的代码后，将上面的代码保存为“polyfits.m”文件。

step 2 使用上面的程序代码，对基础数据进行LS多项式拟合。在MATLAB的命令窗口中输入下面的程序代码：

```
>>x=[ -3.0 -2.0 -1.0 0 1.0 2.0 3.0] ';
y=[ -0.2774 0.8958 -1.5651 3.4565 3.0601 4.8568 3.8982] ';
[ x,i] =sort(x);
y=y(i);
```

```
xi=min(x)+[0:100]/100*(max(x)-min(x));
for i=1:4
N=2*i-1;
[th,err,yi]=polyfits(x,y,N,xi);
subplot(220+i);
plot(x,y,'k*')
hold on
plot(xi,yi,'g:','LineWidth',1.5);
title(['The ',num2str(N),'th Polynomial Curve Fitting'])
grid on
end
```

step 3 查看程序代码的结果。输入上面的程序代码后，按"Enter"键，得到的拟合结果如图4.16所示。

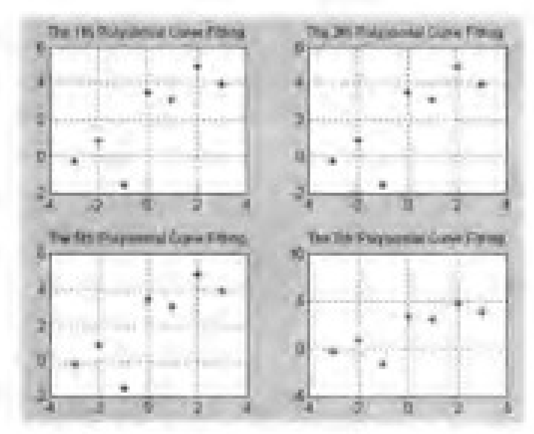


图4.16 使用LS方法求解的拟合结果

step 4 对上面的基础数据，可以使用polyfit命令来进行拟合。在命令窗口中输入下面的程序代码

```
%基础数据
x=[-3.0 -2.0 -1.0 0 1.0 2.0 3.0]';
y=[-0.2774 0.8958 -1.5651 3.4565 3.0601 4.8568 3.8982]';
for i=1:4
N=2*i-1;
[p,s]=polyfit(x,y,N);
yi= polyval(p,x,s);
subplot(220+i);
plot(x,y,'k*')
hold on
plot(x,yi,'g:','LineWidth',1.5);
title(['The ',num2str(N),'th Polynomial Curve Fitting'])
grid on
end
```

step 5 查看程序代码的结果。输入上面的程序代码后，按"Enter"键，得到的拟合结果如图4.17所示。



从上面的例子中可以看出，LS方法其实是WLS方法的一种特例，相当于将每个基础数据的准确度都设为1。但是，自行编写的M文件和默认的命令结果不同，请读者自行仔细比较。

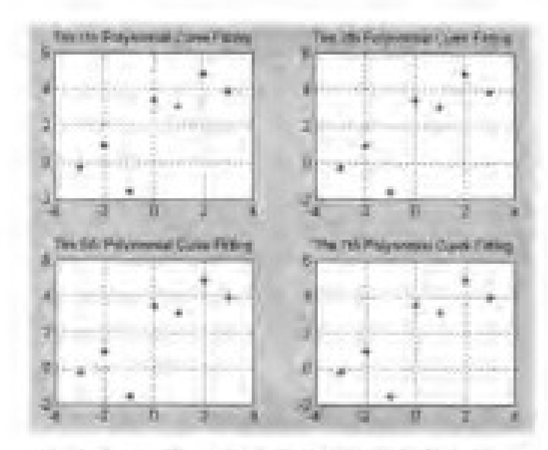


图 4.17 使用默认命令得到的拟合结果

例 4.10 同时使用 WLS 和 LS 方法对基础数据进行拟合，然后画出对比图形。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
clear;
clf;
x=[ 1 3 5 7 9 2 4 6 8 10] ;
y=[ 0.0881 0.9290 2.4932 4.9292 7.9605 ...
    0.9536 2.4836 3.4173 6.3903 10.2443] ;
eb=[ 0.2*ones(5,1);ones(5,1)] ;
[ x,i] =sort(x);
y=y(i);
eb=eb(i);
h=errorbar(x,y,eb,':');
set(h,'LineWidth',1.5);
hold on;
N=2;
xi=[ 0:100] /10;
[ thl,errl,yl] =polyfits(x,y,N,xi);
[ thwl,errwl,ywl] =polyfits(x,y,N,xi,eb);
plot(xi,yl,'g','LineWidth',1.5),
hold on
plot(xi,ywl,'r','LineWidth',1.5)
grid
axis([ 0 10.5 -1 11.5] )
title('WLS VS LS');
legend('Original','LS','WLS')
box on
```

step 2 查看程序代码的结果。当输入上面的程序代码后，按 "Enter" 键，查看拟合的结果如图 4.18 所示。



说明

从上面的图形结果中可以看出，使用 WLS 得到的拟合结果比 LS 得到的拟合结果更加准确，只是参数比 LS 的参数更多。

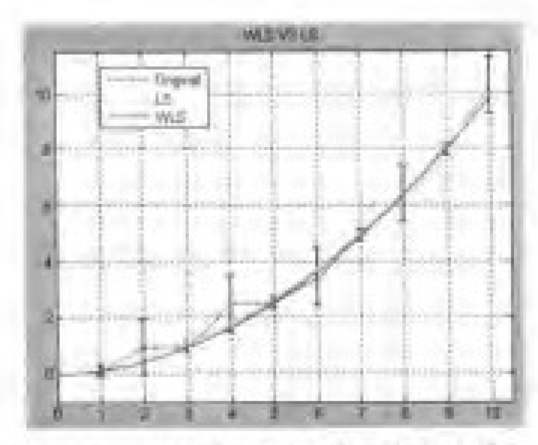


图 4.18　WLS 和 LS 方法得到的拟合结果

4.3 曲线拟合图形界面

在MATLAB7.0中，为用户提供曲线拟合图形界面，用户可以在该界面上直接进行曲线拟合。在该界面中，用户可以实现 Spline 曲线拟合，也可以对同一组数据系列使用多种拟合方法、绘制拟合残余等多种功能。最后，该界面还可以将拟合结果和估计数值保存到 MATLAB 的工作空间中。

4.3.1 曲线拟合

在本小节中，将以一个简单的例子来演示如何使用该界面完成拟合工作。

例 4.11　以例 4.10 中的基础数据为例，使用曲线拟合图形界面完成各种拟合工作，下面分步骤详细介绍。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> x=[ 1 3 5 7 9 2 4 6 8 10] ;
>>y=[ 0.0881 0.9290 2.4932 4.9292 7.9605 ...
    0.9536 2.4836 3.4173 6.3903 10.2443] ;
>>plot(x,y,'ro')
>> grid
```

step 2 查看程序代码的结果。输入代码后，按 "Enter" 键，得到的图形如图 4.19 所示。

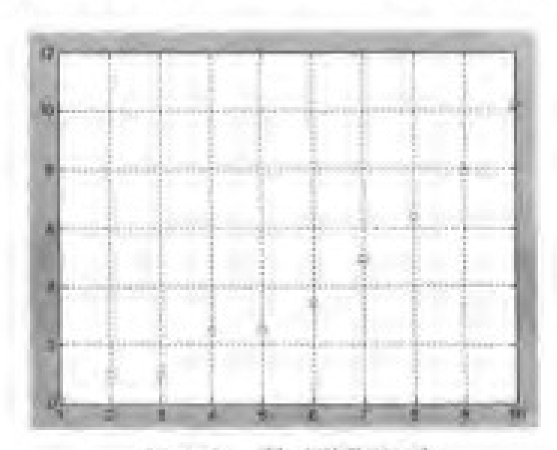


图 4.19　基础数据图形

step 3 打开"Basic Fitting-1"对话框。选择图形窗口中的"Tools"⇨"Basic Fitting"命令，打开关于该数据的拟合图形界面，如图 4.20 所示。

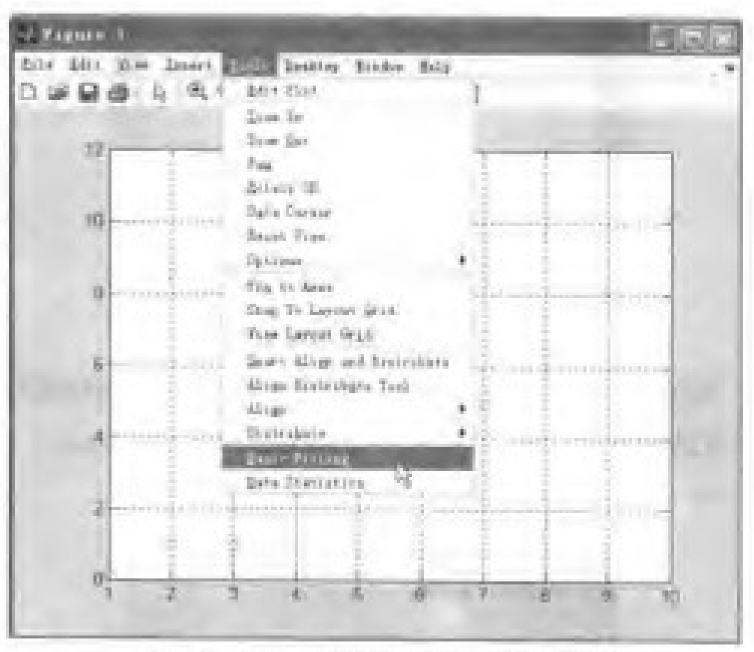


图 4.20 打开"Basic Fitting-1"对话框

step 4 查看对话框。当选择上面的菜单选项后，就可以打开默认情况下的对话框，如图 4.21 所示。



图 4.21 默认的"Basic Fitting-1"对话框

step 5 查看图形的变化。MATLAB在打开拟合图形的对话框的同时，还会为原始的图形添加图例，如图 4.22 所示。



在MATLAB中，会将原始的基础数据名称设置为"data1"，可以直接在图形窗口中修改这个名称。在本实例中，为了简化步骤，保存默认的名称。

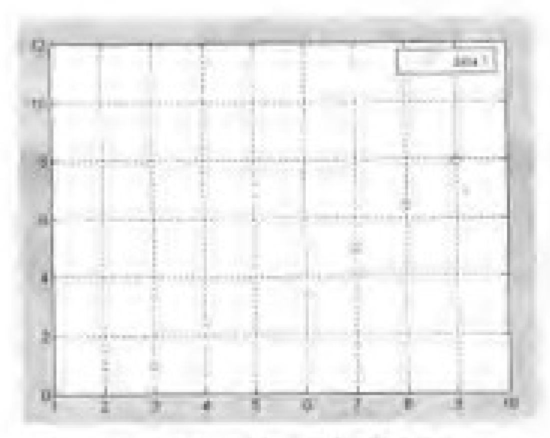


图 4.22　添加图例后的图形

step 6 展开对话框，选择数据拟合的类型，同时在图形中显示拟合的结果方程。将默认的"Basic Fitting-1"对话框展开，在"Check to display fits on figure"选框中选择"cubic"，然后选中"Show equation"选项，如图 4.23 所示。

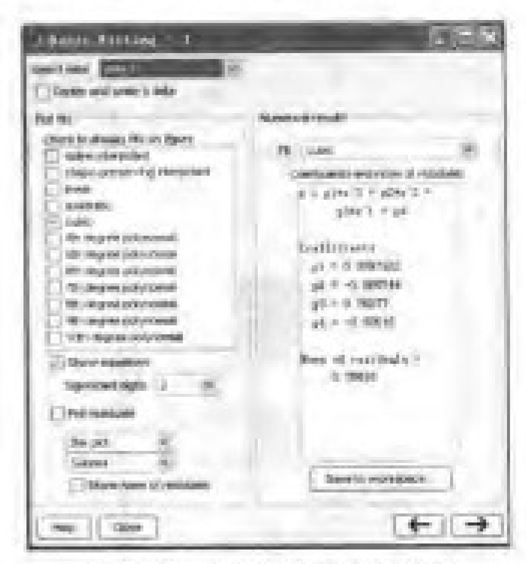


图 4.23　选择数据拟合的类型

step 7 查看图形的变化。当选择上面的选项后，原始图形会显示拟合图形和方程，如图 4.24 所示。

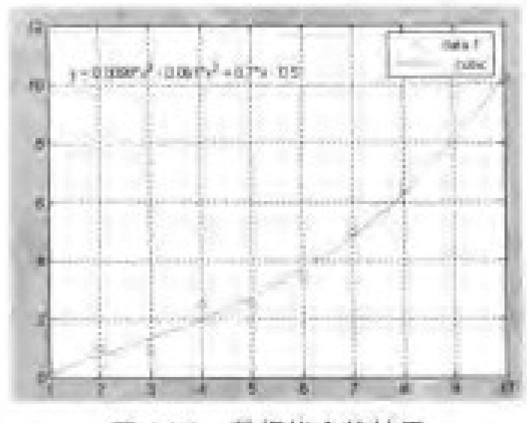


图 4.24　数据拟合的结果



如果选中对话框中的"Center and scale X data"选项，MATLAB会将基础数据拟合为平均值为0，单位标准方差的数据。并不是所有的数据都适合上面的方法，当系统检测出用户使用该方法在不合适的位置时，显示对应的警告信息。

4.3.2 绘制拟合残差图形

延续上面小节的步骤。

step 1 绘制拟合残差图形，并显示残差的标准差。选择"Basic Fitting-1"对话框中的"Plot residuals"选项和"Show norm of residuals"选项，如图 4.25 所示。



图 4.25 显示拟合残差以及其标准差

step 2 查看绘制的结果。当选择上面的选项后，MATLAB 会在原始图形的下方绘制残差图形，并在图形中显示残差的标准差，得到的结果如图 4.26 所示。

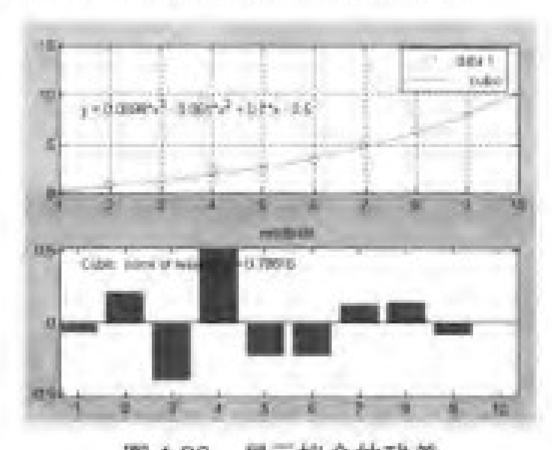


图 4.26 显示拟合的残差



在"Basic Fitting-1"对话框中，可以选择残差图形的图表类型：Bar、Scatter和Line，可以在对应的选项框中选择图表类型；同时，可以选择绘制残差图形的位置：Subplot和Separate Figure两个选项。在默认情况下，图表类型为Bar，位置为Subplot。

step 3 保存拟合的结果。单击"Basic Fitting-1"对话框中的"Save to workspace"按钮，打开

"Save Fit to workspace" 对话框，在其中设置保存选项，如图 4.27 所示。



图 4.27 保存拟合的结果

step 4 查看保存结果。当保存上面的拟合结果后，返回 MATLAB 的命令窗口中，查看保存的结果：

```
Variables have been created in the current workspace.
>> fit1
fit1 =
      type: 'polynomial degree 3'
     coeff: [0.0098 -0.0607 0.7028 -0.5002]
>> normresid1
normresid1 =
    0.7882
```

4.3.3 进行数据预测

延续上面小节的步骤。

step 1 对数据进行预测。再次展开 "Basic Fitting-1" 对话框，打开 "Find Y = f(X)" 面板，在 "Enter value(s)" 选框中输入 "10:15"，然后单击 "Evaluate" 按钮，在其下面的选框中会显示预测的数据。最后，选择对话框中的 "Plot evaluated results" 选项，将预测的结果显示在图形中，如图 4.28 所示。



图 4.28 预测数据

step 2 查看绘制结果。当选择上面选项后，MATLAB会在原来图形基础上显示预测的数据，如图4.29所示。

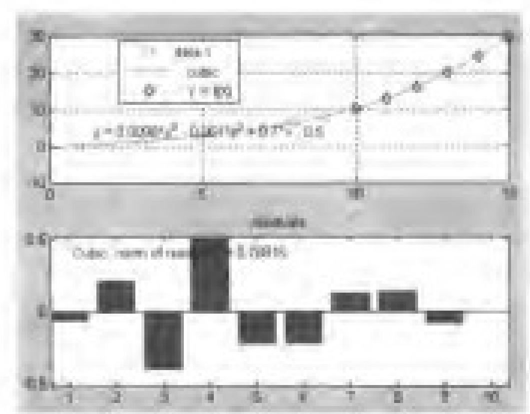


图 4.29　显示预测数据的图形



在"Find Y = f(X)"面板的"Enter value(s)"选框中，可以输入任何合理的MATLAB表达式，例如已经定义过的变量，或者数学表达式等。

step 3 保存预测的数据。单击"Find Y = f(X)"面板中的"Save to workspace"按钮，打开"Save Results to workspace"对话框，在其中设置保存数据选项，然后单击"Save Results to workspace"对话框中的"OK"按钮，保存预测的数据，如图4.30所示。



图 4.30　保存预测数据

step 4 查看保存结果。当设置保存选项后，返回MATLAB的命令窗口中，查看保存的结果：

```
Variables have been created in the current workspace.
>> [ x1 fx]
ans =
   10.0000   10.2464
   11.0000   12.9151
   12.0000   16.1087
   13.0000   19.8859
```

```
14.0000    24.3055
15.0000    29.4262
```



上面的例子比较简单，基本演示了如何使用曲线拟合曲线界面的方法，读者可以根据实际情况，选择不同的拟合参数，完成其他的拟合工作。

4.4 傅里叶分析

傅里叶（Fourier）分析在信号处理领域有着广泛应用，现实生活中大部分信号都包含多个不同频率组件，这些信号组件频率会随着时间或快/或慢的变化。傅里叶（Fourier）系列和傅里叶变换是用来分析周期或者非周期信号的频率特性的数学工具。从时间的角度来看，傅里叶分析包括连续时间和离散时间的Fourier变换，总共有四种不同的Fourier分析类型：连续时间的Fourier系列、连续时间的Fourier变换、离散时间的Fourier变换和离散Fourier系列/变换等。其中，离散Fourier系列/变换是最容易被程序化的，因此本节将详细叙述离散Fourier系列和变换的内容。

4.4.1 离散 Fourier 变换

在信号领域中，离散Fourier变换的定义如下：从某连续时间信号中每Ts根据$\{x[n=x](nT), n=0:M-1\}$关系选取信号数据系列。

该数据系列的离散Fourier变换（DFT）和逆离散Fourier变换（IDFT）的定义为：

$$\text{DFT}: x(k)=\sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N} \qquad k=0,\cdots,N-1$$

$$\text{IDFT}: x[n]=\frac{1}{N}\sum_{n=0}^{N-1} x(k)e^{j2\pi nk/N} \qquad k=0,\cdots,N-1$$

关于上面的两种Fourier变换，MATLAB提供了FFT和IFFT命令来求解。FFT是指快速Fourier变换，使用快速的算法来计算上面两种Fourier变换。其相应的调用命令如下：

- Y = fft(X,n,dim)
- y = ifft(X,n,dim)

在上面的命令中，n表示离散系列的次序编号，X则表示离散信号序列。



关于FFT命令的原理，由于比较复杂，在这里就不详细展开了，感兴趣的读者可以查阅相应的书籍。

例 4.12 使用FFT，从包含噪声信号在内的信号信息中寻找组成信号的主要频率。

step 1 产生原始信号，并绘制信号图形。在MATLAB命令窗口中输入下面的程序代码：

```
>> t = 0:0.001:0.6;
>>x = sin(2*pi*50*t)+sin(2*pi*120*t);
>>y = x + 2*randn(size(t));
>>plot(1000*t(1:50),y(1:50))
>>title('Signal Corrupted with Zero-Mean Random Noise')
```

```
>>xlabel('time (milliseconds)')
>> grid
```

step 2 查看原始信号的图形。在输入上面代码后，按"Enter"键，得到的原始图形如图4.31所示。

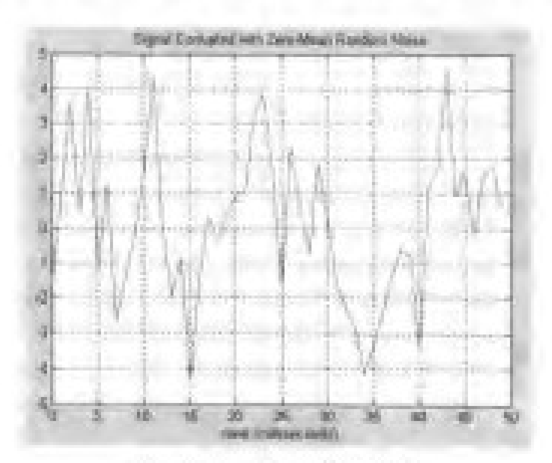


图4.31 原始噪声信号



上面程序代码产生的信号中包含主频率为50 Hz和120 Hz，同时信号中包含均值为0的随机噪声信号。之所以这样设置信号属性，是为了在后面的步骤中进行频率分析。

step 3 对信号进Fourier变换。在MATLAB命令窗口中输入下面的程序代码：

```
%进行Fourier变换
>> Y = fft(y,512);
>> Pyy = Y.* conj(Y) / 512;
>> f = 1000*(0:256)/512;
%打开新的图形窗口
>> figure
>> plot(f,Pyy(1:257))
>> title('Frequency content of y')
>> xlabel('frequency (Hz)')
>> grid
```

step 4 查看信号转换图形。在输入上面代码后，按"Enter"键，得到的信号转换图形如图4.32所示。

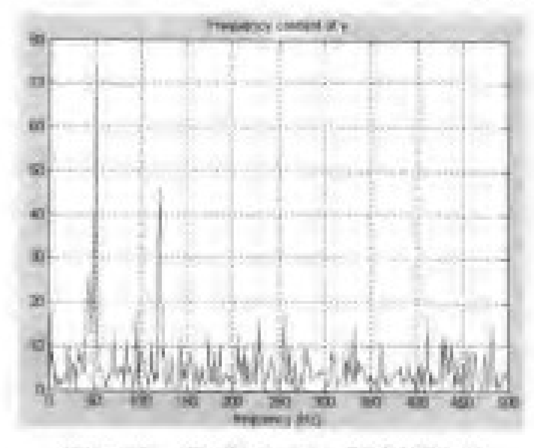


图4.32 经过Fourier变换的信号



从上面的Fourier变换结果可以看出，在频率为50 Hz和120 Hz的地方有较强的信号，而在其他地方则没有明显的信号信息。因此，通过Fourier变换可以从信号中区分频率分布。

4.4.2 FFT和DFT

前面曾经提到过，MATLAB提供FFT函数命令来实现离散Fourier变换（DFT），该命令对应的是快速计算算法。为了让读者更加直观地了解到FFT命令算法相对于DFT算法的优势，在本小节中，将使用一个简单的例子，分别使用FFT和DFT方法来进行Fourier变换，比较两者的优劣。

例4.13　分别使用FFT和DFT方法实现Fourier变换，并比较两者的优劣。

step 1 在MATLAB命令窗口中输入下面的程序代码：

```
clear;
clf
N=2^10;
n=[0:N-1];
x=cos(2*pi*200/N*n)+0.5*sin(2*pi*300/N*n);
tic
%使用DFT方法
for k=0:N-1
    X(k+1)=x*exp(-j*2*pi*k*n/N).';
end
    k=[0:N-1];
%使用IDFT方法
    for n=0:N-1
        xr(n+1)=X*exp(j*2*pi*k*n/N).';
    end
        time_dft=toc;
        subplot(211)
        plot(k,abs(X),'g')
        axis([0 1025 0 600])
        title('DFT Method')
        grid
        hold on
        tic
%使用FFT方法
        X1=fft(x);
%使用IFFT方法
        xr1=ifft(X1);
        time_fft=toc;
        subplot(212)
        plot(k,abs(X1),'r')
        axis([0 1025 0 600])
        title('FFT Method')
        grid
```

step 2 查看程序结果。在输入程序代码后，按"Enter"键，得到的结果如图4.33所示。

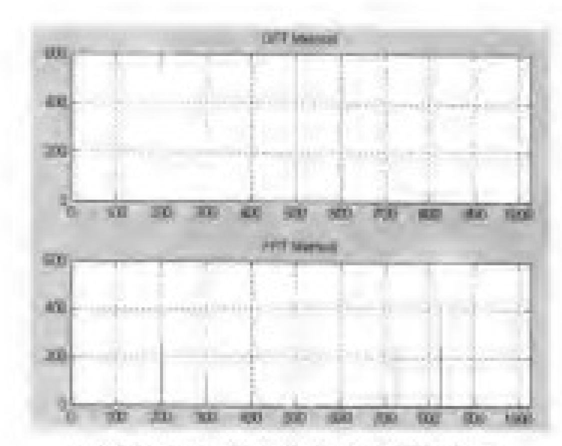


图 4.33　两种变换方法得到的结果

step 3 比较两个方法的计算时间。在 MATLAB 命令窗口中输入下面的程序代码：

```
>> s1 = ['The time of DFT Method is ' num2str(time_dft)];
>>s2 = ['The time of FFT Method is ' num2str(time_fft)];
>>S = strvcat(s1,s2);
>>disp(S)
```

step 4 查看程序结果。在输入程序代码后，按 "Enter" 键，得到的结果如下：

```
The time of DFT Method is 1.742
The time of FFT Method is 0.01
```

从上面的结果可以看出，使用DFT完成Fourier变换的时间是1.742s，而使用FFT方法完成Fourier变换的时间是0.01s。



从上面的实例中可以看出，两种方法得到的Fourier变换结果相同，但是消耗的时间和资源是完全不同的，FFT 方法明显优于 DFT 方法。

4.4.3 DFT 的物理含义

由于DFT在公式表达式上比较复杂，因此读者也许不能理解DFT的物理含义。为了让读者更加容易理解DFT的物理含义，本小节从一个连续信号中选择不同的时间间隔的离散信号，并对各种离散信号进行 DFT 转换。

其中连续信号的表达式如下：

$$x(t)=\sin(1.5\pi t)+0.5\cos(3\pi t)$$

在本小节中，将从这个连续信号中选择不同周期的离散信号，并对其进行离散Fourier变换，最后绘制各种信号图形和转换图形。

例 4.14　从上面的连续信号表达式中选择不同间隔的离散信号，然后进行离散 Fourier 变换，最后绘制各种图形。

step 1 在 MATLAB 命令窗口中输入下面的程序代码：

```
clear;
clf
w1=1.5*pi;
```

```
w2=3*pi;
N=32;
n=[0:N-1];
T=0.1;
t=n*T;
xan=sin(w1*t)+0.5*sin(w2*t);
subplot(421)
stem(t,xan,'.');
axis tight
grid
k=0:N-1;
Xa=fft(xan);
dscrp=norm(xan-real(ifft(Xa)));
subplot(423)
stem(k,abs(Xa),'.');
axis tight
grid

N=32;
n=[0:N-1];
T=0.05;
t=n*T;
xbn=sin(w1*t)+0.5*sin(w2*t);
subplot(422)
stem(t,xbn,'.');
axis tight
grid
k=0:N-1;
Xb=fft(xbn);
subplot(424)
stem(k,abs(Xb),'.');
axis tight
grid

N=64;
n=[0:N-1];
T=0.1;
t=n*T;
xcn=sin(w1*t)+0.5*sin(w2*t);
subplot(425)
stem(t,xcn,'.');
axis tight
grid
k=0:N-1;
Xc=fft(xcn);
subplot(427)
stem(k,abs(Xc),'.');
axis tight
grid

N=64;
n=[0:N-1];
T=0.05;
t=n*T;
xdn=sin(w1*t)+0.5*sin(w2*t);
subplot(426)
```

```
stem(t,xdn,'.');
axis tight
grid
k=0:N-1;
Xd=fft(xdn);
subplot(428)
stem(k,abs(Xd),'.');
axis tight
grid
```

step 2 查看程序代码结果。当输入上面的程序代码后，按"Enter"键，得到的结果如图4.34所示。

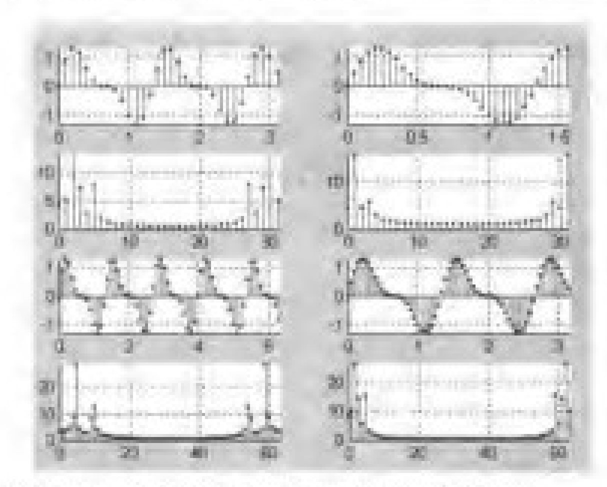


图 4.34 离散信号和 Fourier 转换后的图形



从上面的图形结果中，对比离散数据点的选择个数对离散信号和Fourier变换的影响，从图形中可以形象地看出Fourier变换的物理含义。

4.4.4 使用DFS进行插值

MATLAB提供一维快速Fourier变换(FFT)进行插值的内置函数interpft，该函数主要通过Fourier变换将输入的数据变换到频域，然后使用更多点的Fourier变换，变换到时域，因此其结果就是对数据增加采样，进行插值。

interpft函数的主要调用格式如下：

- y = interpft(x,n)
- y = interpft(x,n,dim)

在上面的命令中，x表示需要进行插值的数据系列，n表示希望采用n点傅里叶逆变换变回到时域中，dim则表示这种插值需要在指定的维度上进行。



由于上面的命令比较简单，关于该命令的使用方法和具体实例，请读者自行查阅相应的帮助文件，这里就不详细介绍了。

在本小节中，将主要介绍如何使用DFS来进行数据插值。要使用DFS来进行数据插值，首先需要

从信号序列$X(k)$中选择数据点，并使用下面的公式进行插值运算：

$$\tilde{x}(t)=\frac{1}{N}\sum_{|k|<N/2}\tilde{X}(k)e^{j2\pi kt/NT}$$

$$=\frac{1}{N}\{X(0)+2\sum_{k=1}^{N/2-1}Real\{X(k)e^{j2\pi kt/NT}\}+X(N/2)\cos(\pi t/T)\}$$

根据上面的公式，读者可以自行编写 M 文件，然后使用编写的文件进行插值运算。

例 4.15 根据上面的基础插值公式，自行编写使用 DFS 方法进行插值的 M 文件，然后以 $f(t)=\sin(\pi t)+0.5\sin(0.5\pi t)$为主信号，以$z(t)=rand(1,n)-0.5$为噪声信号为数据信号源，产生基础数据，最后使用 DFS 进行插值。

step 1 选择命令窗口编辑栏中的“File” ⇨ “New” ⇨ “M-File” 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function [xi,yi]=interp_DFS(T,x,Ws,ti)
%T: 样本间隔
%x: 离散时间样本
%Ws:归一化的频率
%ti: 插值的时间系列

if nargin<4
    ti=5;
end
if nargin<3 | Ws>1
    Ws=1;
end
N=length(x);
if length(ti)==1
    ti=0:T/ti:(N-1)*T; %使用 ti 得到的细化数据系列
end
ks=ceil(Ws*N/2);
yi=fft(x);
yi(ks+2:N-ks)=zeros(1,N-2*ks-1);     %筛选后的时间系列
xi=zeros(1,length(ti));
for k=2:N/2
    xi=xi+yi(k)*exp(j*2*pi*(k-1)*ti/N/T);
end
xi=real(2*xi+yi(1)+yi(N/2+1)*cos(pi*ti/T))/N;   %根据公式计算插值结果
```

在输入上面的程序代码后，将上面的程序代码保存为“interp_DFS.m”文件，在后面的步骤中，将会使用该函数进行数据插值。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
T=0.1;
N=32;
ti=[0:T/5:(N-1)*T];
w1=pi;
w2=0.5*pi;
n=[0:N-1];
t=n*T;
x=sin(w1*t)+0.5*sin(w2*t)+rand(1,N)-0.5;
%绘制原始数据图形
```

```
subplot(411)
plot(t,x,'k.');
title('Original and interpolation signal')
grid on;
axis tight
hold on;
%进行DFS数据插值
[xi,yi]=interp_DFS(T,x,1,ti);
plot(ti,xi,'r')
k=[0:N-1];
%绘制spectrum图形
subplot(412)
stem(k,abs(yi),'k.');
axis tight
title('Original spectrum')
box on
%改变参数，重新进行DFS数据插值
[xi,yi]=interp_DFS(T,x,1/2,ti);
subplot(413)
stem(k,abs(yi),'r.');
title('filtered spectrum')
axis tight
box on
%绘制插值的图形
subplot(414)
plot(t,x,'k.',ti,xi,'r')
title('filtered/smoothed signal')
grid
axis tight
```

step 3 查看程序代码结果。输入程序代码后，按"Enter"键，得到结果如图 4.35 所示。

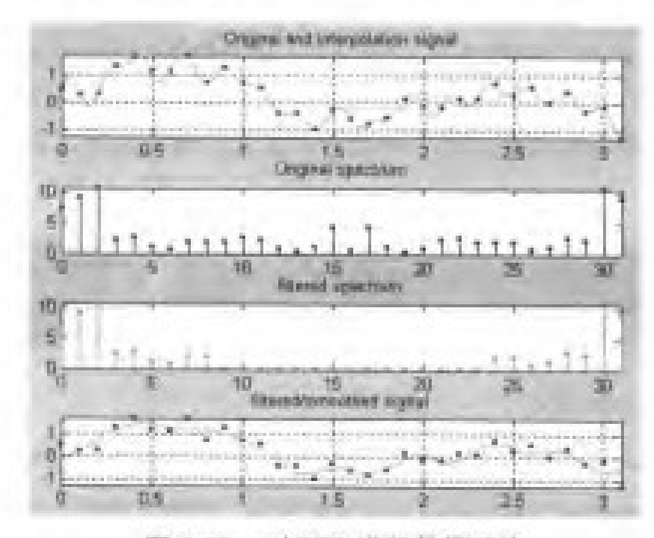


图 4.35　对 DFS 进行数据插值



在上面的例子中，首先根据公式编写关于DFS数据插值的M文件程序代码，然后使用不同的精度对基础信号进行数据插值。读者可以仔细查看程序代码，有助于对DFS数据插值原理的理解。

4.5 优化

在科学工程领域中，优化有着十分广泛的应用。根据数学理论定义，优化是指在某种约束条件下，寻求目标函数的最大值或者最小值。将上面的定义转换为数学公式：

$$\min_{x \in S} \quad f(x)$$

在上面的公式中，x表示变量向量，也就是$x=(x_1, x_2, \cdots x_n)$；$f(x)$也就是优化情况下的目标函数，S表示的是所承受的约束条件。如果没有接受任何的约束条件，或者S是数值条件下的全集，则该优化为非约束优化；否则，则是约束优化。本节首先介绍非约束优化的内容，然后介绍约束优化的内容，最后，将介绍线性规划和二次规划这两种在实际中比较常见的优化问题。

本节所涉及的内容将都是MATLAB内置的函数，有些比较复杂的优化处理工具将会涉及Optimization Toolbox中的函数和内容。如果读者希望自行演示本节中的程序代码，请选择安装Optimization Toolbox组件。

4.5.1 无约束非线性优化命令

在上面介绍过，无约束优化相当于约束集为全集。MATLAB为解决非约束优化提供fminsearch和fminunc两种函数，其对应的详细调用格式如下：

```
[x,fval,exitflag,output] = fminsearch(fun,x0,options)
```

在上面的命令格式中，参数比较繁多，下面分别分详细介绍。

- **输入参数：** 参数"fun"表示优化的目标函数，参数表示执行优化的初始数值，参数"options"表示进行优化的各种属性，一般需要使用optimset函数来进行设置。
- **输出参数：** 参数x表示最优解，fval表示最优解对应的函数数值；参数"exitflag"则表示函数退出优化运算的原因，取值为1、0和-1。其中数值1表示函数收敛于最优解，0则表示函数迭代次数超过了优化属性的设置，-1则表示优化迭代算法被output函数终止；参数"output"是一种结构体变量，显示的是关于优化的属性信息，例如优化迭代次数和优化算法等。



在MATLAB中，fminsearch一般适用于没有约束条件的非线性优化情况，对于线性优化的情况，将在后面的章节中详细介绍。

fminunc函数的调用格式如下：

```
[x,fval,exitflag,output,grad,hessian] = fminunc(fun,x0,options)
```

- 该函数的大部分参数的含义和fminsearch函数相同，而输出参数"grad"表示的是函数在最优解处的梯度；参数"hessian"则表示目标函数在最优解的Hessian矩阵数值；参数"exitflag"表示的也是停止最优求解的类型，但是其取值包括了-2、-1，0，1，2和3。因此，该函数比上面的函数能够更加详细地描述优化情况。关于其具体的含义，请自行查看相应的帮助文件。

4.5.2 无约束非线性优化实例

在本小节中，将介绍如何使用这两种比较常见的函数来求解优化问题。

例 4.16 求解二元函数 $f(x)=3x_1^2+2x_1x_2+x_2^2$ 在全集范围之内的最小值，分别使用不同的优化函数和优化属性。为了让读者能够直观地查看优化求解情况，可以在求解之后绘制二元函数的图形。

step 1 选择命令窗口编辑栏中的"File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function [ f,g] = optfun(x)
f = 3*x(1)^2 + 2*x(1)*x(2) + x(2)^2;
if nargout > 1
   g(1) = 6*x(1)+2*x(2);
   g(2) = 2*x(1)+2*x(2);
end
```

在上面的程序代码中，g 表示的是 f 函数的偏导数，满足下面的方程组：

$$\begin{cases} g(1)=\dfrac{\partial f(x)}{\partial x_1}=6x_1+2x_2 \\ g(2)=\dfrac{\partial f(x)}{\partial x_2}=2x_1+2x_2 \end{cases}$$

在输入完上面的程序代码后，将该代码保存为"optfun.m"文件。

step 2 选择优化的初始数值[1,1]，分别使用不同的函数求解优化。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> x0=[ 1,1] ;
>> options = optimset('Display','iter','TolFun',1e-18,'GradObj','on');
>> [ x,fval,exitflag,output,grad] =fminunc(@optfun,x0,options)
                                         Norm of      First-order
 Iteration           f(x)             step          optimality   CG-iterations
     0                   6                                 8
     1         6.28624e-031         1.41421       1.55e-015             1
     2         2.91704e-062     8.9509e-016       5.92e-031             1
Optimization terminated: first-order optimality less than OPTIONS.TolFun,
 and no negative/zero curvature detected in trust region model.
x =
  1.0e-031 *
   -0.9861         0
fval =
  2.9170e-062
exitflag =
     1
output =
       iterations: 2
        funcCount: 3
      cgiterations: 2
     firstorderopt: 5.9165e-031
        algorithm: 'large-scale: trust-region Newton'
          message: [ 1x137 char]
grad =
  1.0e-030 *
   -0.5916
   -0.1972
>> [ x1,fval1,exitflag1,output1] = fminsearch(@optfun,x0,options)
  Iteration   Func-count     min f(x)         Procedure
```

```
     0           1                 6
     1           3                 6         initial simplex
     2           5           5.52062         expand
     3           7           4.91391         expand
     4           9           3.86566         expand
     5          11           2.52517         expand
  .........................................//限于篇幅，省略了部分数据
    78         151       1.95216e-018        contract inside
    79         153       4.03648e-019        contract inside
    80         155       4.03648e-019        contract inside
    81         157       4.03648e-019        contract inside
x1 =
  1.0e-009 *
   -0.4052    0.1308
fval1 =
   4.0365e-019
exitflag1 =
     1
output1 =
    iterations: 81
     funcCount: 157
     algorithm: 'Nelder-Mead simplex direct search'
       message: [1x196 char]
```



在上面的结果中，使用fminuc函数求解的最优解为（0,0），而且迭代的次数为2，对应的优化求解方法为"large-scale: trust-region Newton"；使用fminsearch函数求解的最优解为（0,0），且迭代次数为81，使用的优化求解方法为"Nelder-Mead simplex direct search"。因此，在相同的初值条件下，两个方法求解的性质不同。

step 3 选择优化的初始数值[-1,1]，分别使用不同的函数求解优化。在MATLAB的命令窗口中输入下面的程序代码：

```
>> x0=[-1,1];
>> options = optimset('Display','iter','TolFun',1e-18,'GradObj','on');
>> [x,fval,exitflag,output,grad] =fminunc(@optfun,x0,options)
                                  Norm of      First-order
 Iteration         f(x)           step          optimality   CG-iterations
     0                2                                  4
     1          0.666667       0.666667               1.33            1
     2          0.222222       0.666667               1.33            1
     3         0.0740741       0.222222              0.444            1
     4         0.0246914       0.222222              0.444            1
     5        0.00823045      0.0740741              0.148            1
  .............................................../限于篇幅，省略部分数据
    25      2.36047e-012   1.25445e-006          2.51e-006            1
    26      7.86824e-013   1.25445e-006          2.51e-006            1
    27      2.62275e-013    4.1815e-007          8.36e-007            1
Optimization terminated: Norm of the current step is less than OPTIONS.
TolX.
x =
  1.0e-006 *
   -0.2091    0.6272
fval =
  2.6227e-013
```

```
exitflag =
     2
output =
        iterations: 27
         funcCount: 28
      cgiterations: 27
     firstorderopt: 8.3630e-007
         algorithm: 'large-scale: trust-region Newton'
           message: [1x76 char]
grad =
  1.0e-006 *
    0.0000
    0.8363
>> [x1,fval1,exitflag1,output1] = fminsearch(@optfun,x0,options)
  Iteration   Func-count     min f(x)         Procedure
     0            1              2
     1            3              2          initial simplex
     2            5        1.65062          expand
     3            7        1.47141          expand
     4            9        1.03754          expand
     5           11       0.766885          expand
     6           13       0.766885          contract outside
     7           15       0.766885          contract outside
     8           17       0.602196          expand
     9           18       0.602196          reflect
10           20         0.334623         expand

    ...........................................//限于篇幅，省略了部分数据
    74          143     3.83172e-019          contract outside
    75          145     2.24524e-019          contract inside
    76          147     2.24524e-019          contract inside
x1 =
  1.0e-009 *
   -0.3318    0.3981
fval1 =
  2.2452e-019
exitflag1 =
     1
output1 =
    iterations: 76
     funcCount: 147
     algorithm: 'Nelder-Mead simplex direct search'
       message: [1x196 char]
```

从上面的程序结果中可以看出，当修改优化的初始条件后，各种优化函数所使用的迭代次数会有明显的改变。因此，设置初值条件将直接影响优化求解的效率。



在使用不同函数来求解函数的优化条件时，多次使用optimset函数来设置优化属性。关于该函数的具体使用方法，请自行参看帮助文件。

step 4 在[-1.5, 1.5]范围内绘制二元函数 $f(x)=3x_1^2+2x_1x_2+x_2^2$ 的图形，并从图形中显示对应的函数数值。在 MATLAB 的命令窗口中输入下面的程序代码。

```
>> x=-1.5:0.02:1.5;
>>y=x;
>>[X,Y]=meshgrid(x,y);
>>Z=3*X.^2+2*X.*Y+Y.^2;
>>surf(X,Y,Z)
>>shading interp
>> colorbar horiz
```

step 5 查看函数图形。在输入代码后，按"Enter"键，查看函数图形，如图4.36所示。

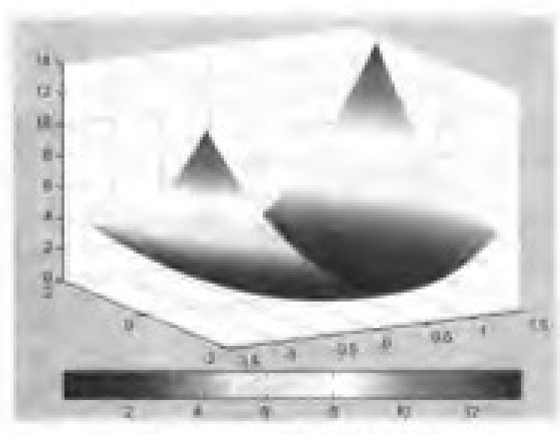


图4.36 三维函数图形



从上面的图形结果中可以看出，(0，0)是该函数在全局范围内的最小值，通过上面的函数都可以求解到该最优解。

4.5.3 非线性最小方差命令

非线性最小方差（nonlinear least-squares）是非线性约束优化的一种特例，其优化的目标函数为 $\text{Min}\sum_{n=1}^{k} f_n^2(x)$ 。在上面的函数表达式中，$f_n^2(x)$ 就是所谓的方差函数。如果熟悉前面介绍的曲线拟合内容，就会发现该目标函数就是曲线拟合的函数。因此，上面的问题也被称为非线性数据拟合（nonlinear data-fitting）。

在MATLAB中，为了求解非线性最小方差（nonlinear least-squares），提供lsqnonlin函数命令，其最完整的调用格式如下：

```
[x,resnorm,residual,exitflag,output,lambda] = lsqnonlin(fun,x0,lb,ub,options)
```

在上面的命令中，函数的参数比较复杂，下面详细介绍各参数的具体含义。

- **输入参数：** 参数fun表示优化的目标函数，参数x0表示优化的初值条件，lb是进行优化求解的下限，ub是进行优化求解的上限，相当于 lb ≤ x ≤ ub；参数options则表示进行优化求解的优化属性。
- **输出参数：** 参数x表示所求得的最优解；参数resnorm则表示二阶范数，在数值上等于sum(fun(x).^2)；参数residual则表示优化求解后的残数；参数lambda则表示函数在最优解处的拉格朗日数值；其他参数和其他优化命令中的含义相同。



在上面的命令格式中，并不是所有的参数都是必须输入的。在输入参数部分，只有fun，x0是必须的，而在输出参数部分，则只有x是必须的。MATLAB之所以提供上面的许多参数，是为了方便用户自行研究和比较优化的属性。

4.5.4 非线性最小方差实例

例 4.17 以函数 $f(x)=\frac{1}{1+8x^2}$ 为基准函数，对其进行非线性最小方差运算，同时以最小方差为目标进行数据拟合。

step 1 选择命令窗口编辑栏中的"File"⇨"New"⇨"M-File"命令，打开M文件编辑器，在其中输入下面的程序代码：

```
function F=flq(a)
xx=-2+[0:200]/50;
F=polyval(a,xx)-1./(1+8*xx.*xx);
```

在输入上面的程序代码后，将上面的程序代码保存为"flq.m"文件。

step 2 进行非线性最小方差求解。在MATLAB的命令窗口中输入下面的程序代码：

```
N=5;
a0=zeros(1,N);
lb=-1*ones(1,N);
ub=ones(1,N);
options = optimset('Display','iter','TolFun',1e-18,'GradObj','on');
[x,resnorm,residual,exitflag,output,lambda] = lsqnonlin('flq',a0,lb,ub,
options);
```

step 3 查看程序结果。在输入上面的程序代码后，按"Enter"键，得到的结果如下：

```
                                          Norm of      First-order
 Iteration  Func-count     f(x)           step       optimality   CG-iterations
     0          6        27.7062                         49.4
     1         12        12.2685        0.261626         9.63              2
     2         18        9.81249         0.16763         18.1              2
     3         24        6.76791        0.195781         4.87              2
     4         30        5.94883        0.105258         7.67              2
     5         36        4.93127        0.149942         3.48              2
  .......................................................// 省略了部分数据
    32        198        3.89864    3.62219e-006       6.6e-005            2
    33        204        3.89864    2.94793e-006      8.93e-005            2
    34        210        3.89864    1.63203e-006      6.33e-005            2
    35        216        3.89864    2.08912e-006      7.21e-005            2
    36        222        3.89864    9.19467e-007      2.77e-005            2
>> x
x =
    0.1124    0.0000   -0.5522    0.0000    0.6225
>> resnorm
resnorm =
    3.8986
>> exitflag
exitflag =
```

```
    2
>> output
output =
    firstorderopt: 2.7747e-005
       iterations: 36
        funcCount: 222
     cgiterations: 71
        algorithm: 'large-scale: trust-region reflective Newton'
          message: [1x77 char]
>> lambda.lower
ans =
     0
     0
     0
     0
     0
>> lambda.upper
ans =
     0
     0
     0
     0
     0
```

step 4 分析非线性方差运算的数据残余。在命令窗口中输入下面的程序代码：

```
>> xx=-2+[0:200]/50;
>> plot(xx,residual,'r','LineWidth',1.5)
>> grid
>> title('The residual of data')
```

step 5 查看图形。在输入程序代码后，按"Enter"键，得到的结果如图 4.37 所示。

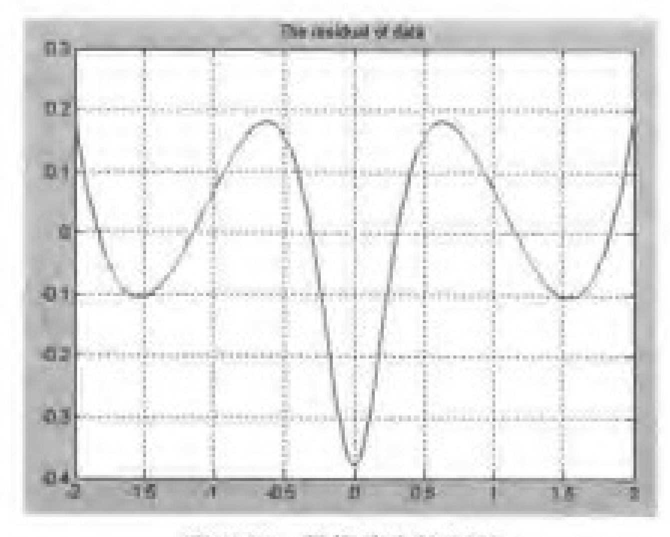


图 4.37　数据残余的图形



限于篇幅，关于该命令中的其他参数在这里就不详细介绍了，感兴趣的读者请自行阅读相应的帮助文件。

4.5.5 约束条件的非线性优化命令

有约束条件的优化情况比无约束条件的优化情况要复杂得多，处理起来也更困难，种类也比较繁杂，限于篇幅，这里不能将所有的约束优化都进行讨论，仅限于讨论MATLAB内置函数fmincon的使用方法。首先，fmincon函数主要用于解决具有下面约束条件的优化

$$\min_x \quad f(x)$$

$$\text{Subject to} \quad c(x) \leqslant 0$$

$$\begin{aligned} &\text{ceq}(x) = 0 \\ &A \cdot x \leqslant b \\ &\text{Aeq} \cdot x = \text{beq} \\ &lb \leqslant x \leqslant ub \end{aligned}$$

该函数的完整调用格式如下：

```
[x,fval,exitflag,output,lambda] = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

该函数的参数比较复杂，下面详细介绍各种参数的含义。

- **输入参数**：参数fun表示的是优化目标函数，x0表示的是优化的初始值，参数A、b表示的是满足线性关系式 $A \cdot x \leqslant b$ 的系数矩阵和结果矩阵；参数Aeq、beq表示的是满足线性等式 $\text{Aeq} \cdot x = \text{beq}$ 的矩阵；参数lb、ub则表示满足参数取值范围 $lb \leqslant x \leqslant ub$ 的上限和下限；参数nonlcon则表示需要参数满足的非线性关系式 $c(x) \leqslant 0$ 和 $\text{ceq}(x)=0$ 的优化情况；参数options就是进行优化的属性设置；
- **输出参数**：exitflag表示程序退出优化运算的类型，取值为-2、-1、0、1、2、3、4和5，其数值对应的类型在这里就不详细展开了；output参数包含多种关于优化的信息，包含iterations、funcCount、algorithm、cgiterations、stepsize和firstorderopt等；参数lambda则表示lower、upper、ubineqlin、eqlin、ineqnonlin和eqnonlin等，分别表示优化问题的各种约束问题的拉格朗日参数数值。



尽管各种优化命令中都包含了上面的一些主要参数，但是在不同的优化命令中，上面参数的含义是不相同的，不要认为相同参数的含义都一致。

4.5.6 约束条件的非线性优化实例

例4.18 求解在约束条件 $0 \leqslant x_1 + 2x_2 + 3x_3 \leqslant 72$ 下，函数 $f(x) = -x_1 x_2 x_3$ 的最小值的最优解以及最优解的数值。

step 1 转换约束条件，将上面的约束条件转换为下面的关系式：

$$\begin{cases} -x_1 - 2x_2 - 3x_3 \leqslant 0 \\ x_1 + 2x_2 + 3x_3 \leqslant 72 \end{cases}$$



由于在fmincon中，所有的约束条件都是以上面的不等式形式出现的，因此在本步骤中需要将原来的约束条件转换为不等式方程组。

step 2 选择命令窗口编辑栏中的“File” ⇨ “New” ⇨ “M-File” 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function  f=optcon(x)
f=-x(1)*x(2)*x(3);
```

将上面的程序代码保存为 “optcon.m” 文件，该文件将是最优解的目标函数。

step 2 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> A=[ 1,-2,-2;1,2,2] ;
>> b=[ 0;72] ;
>> x0=[ 10;10;10] ;
>>[ x,fval,exitflag,output,lambda]  = fmincon(@optcon,x0,A,b);
```

step 4 查看优化信息。在输入上面的程序代码后，按 “Enter” 键，MATLAB 将会进行优化运算，并显示对应的优化信息：

```
Optimization terminated: Magnitude of directional derivative in search
 direction less than 2*cptions.TolFun and maximum constraint violation
  is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower       upper      ineqlin    ineqnonlin
                             2
```

在上面的程序中显示了实质起作用的约束条件和优化中止的类型。

step 5 查看优化的结果。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> x
x =
   24.0000
   12.0000
   12.0000
>> fval
fval =
 -3.4560e+003
>> exitflag
exitflag =
     5
>> output
output =
       iterations: 8
        funcCount: 48
         stepsize: 1
        algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
    firstorderopt: 1.5459e-004
     cgiterations: []
          message: [ 1x172 char]
>> lambda
lambda =
         lower: [ 3x1 double]
         upper: [ 3x1 double]
         eqlin: [ 0x1 double]
      eqnonlin: [ 0x1 double]
```

```
          ineqlin: [ 2x1 double]
       ineqnonlin: [ 0x1 double]
>> lambda.lower
ans =
     0
     0
     0
>> lambda.upper
ans =
     0
     0
     0
```



从上面的优化结果中，可以看出各种具体的优化信息，进而进行优化分析。由于在上面的步骤中，没有显示迭代过程，在该优化结果中的output参数中显示了迭代次数为8。

step 6 重新设置优化条件，进行优化运算。将最优问题的约束条件修改为下面的关系式：

$$\begin{cases} x_2 \leqslant 5 \\ x_3 \leqslant 10 \\ x_1 + 2x_2 + 3x_3 \leqslant 72 \end{cases}$$

同时，将初值设置为[1，1，1]，然后进行优化求解，得到的结果如下：

```
>> x0=[ 1,1,1] ;
>> A=[ 1,2,2;0,1,0;0,0,1] ;
 b=[ 72;5;10] ;
>> [ x,fval,exitflag,output,lambda] = fmincon(@optcon,x0,A,b);
Warning: Large-scale (trust region) method does not currently solve
this type of problem,
 switching to medium-scale (line search).
> In fmincon at 260
Optimization terminated: first-order optimality measure less
 than options.TolFun and maximum constraint violation is less
 than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
                          1
                          2
                          3
>> x
x =
   42.0000    5.0000   10.0000
>> fval
fval =
 -2.1000e+003
>> exitflag
exitflag =
     1
>> output
output =
       iterations: 5
        funcCount: 29
         stepsize: 1
```

```
        algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
    firstorderopt: 0
     cgiterations: []
          message: [1x144 char]
>> lambda
lambda =
         lower: [ 3x1 double]
         upper: [ 3x1 double]
         eqlin: [ 0x1 double]
      eqnonlin: [ 0x1 double]
        ineqlin: [ 3x1 double]
    ineqnonlin: [ 0x1 double]
>> lambda.lower
ans =
     0
     0
     0
>> lambda.upper
ans =
     0
     0
     0
```



从上面的条件中可以看出，当修改关于优化的各种属性后，优化问题会发生质的改变，因此在进行优化求解问题的时候，需要特别注意优化求解的条件。

4.5.7 最小最大值的优化问题

最小最大值的优化问题函是一个比较特殊的问题，其表示的是从一系列最大值中选取最小的数值，相当于求解下面的优化问题：

$$\mathop{\mathrm{Min}}_{x} \quad \max_{\{F_i\}} \quad \{F_i(x)\}$$

$$\text{Subject to} \quad c(x)\leqslant 0$$

$$ceq(x)=0$$
$$A\cdot x\leqslant b$$
$$Aeq\cdot x=beq$$
$$lb\leqslant x\leqslant ub$$

在上面的目标函数中，$F(x)=[f_1(x),f_2(x),\cdots,f_N(x)]^T$。该函数的参数和使用方法和前面介绍的 fmincon 完全相同，这里就不重复介绍了。下面使用简单的实例来介绍如何使用该函数命令。

例 4.19　求解函数 $F(x)=[f_1(x),f_2(x),f_3(x),f_4(x),f_5(x)]$ 的最小最大值，其中各分函数依次为 $f_1(x)=2x_1^2+x_2^2-48x_1-40x_2+125$，$f_2(x)=-x_1^2-3x_2^2$，$f_3(x)=x_1+3x_2-18$，$f_4(x)=-x_1-x_2$ 和 $f_5(x)=x_1+x_2-8$。

step 1 选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function f = mnmax(x)
f(1)= 2*x(1)^2+x(2)^2-48*x(1)-40*x(2)+125;
```

```
f(2)= -x(1)^2 - 3*x(2)^2;
f(3)= x(1) + 3*x(2) -18;
f(4)= -x(1)- x(2);
f(5)= x(1) + x(2) - 8;
```

在输入上面的程序代码后，将该代码保存为“mnmax.m”文件。

step 2 求解最小最大值的优化问题。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> x0 = [0.1; 0.1];
>> [x,fval] = fminimax(@mnmax,x0)
Optimization terminated: Search direction less than 2*options.TolX
 and maximum constraint violation is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
                                      4
                                      5
x =
    1.3333
    2.6667
fval =
  -34.9994  -23.1121   -8.6665   -4.0000   -4.0000
```

在上面的求解过程中，首先设置了初值条件，然后直接调用函数求解优化问题。

step 2 重新设置优化条件，求解优化问题。在命令窗口中输入下面的程序代码：

```
>> x0 = [0.1; 0.1];        % Make a starting guess at solution
options = optimset('MinAbsMax',5);    % Minimize absolute values
[x,fval,maxfval,exitflag,output,lambda] = fminimax(@mnmax,x0,[],[],[],
[],[],[],[],options)
Optimization terminated: Search direction less than 2*options.TolX
 and maximum constraint violation is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
                                      6
                                      7
                                      8
x =
    1.5768
    1.7126
fval =
  -11.2854  -11.2854  -11.2854   -3.2894   -4.7106
maxfval =
   11.2854
exitflag =
     4
output =
       iterations: 9
        funcCount: 49
         stepsize: 1
        algorithm: 'minimax SQP, Quasi-Newton, line_search'
    firstorderopt: []
     cgiterations: []
          message: [1x129 char]
lambda =
```

```
       lower: [2x1 double]
       upper: [2x1 double]
       eqlin: [0x1 double]
    eqnonlin: [0x1 double]
     ineqlin: [0x1 double]
  ineqnonlin: [0x1 double]
```



和其他的优化求解方法相似，当修改优化条件后，相同的优化问题将会得出不同的结果。尽管可以使用optimset来设置各种优化的属性，但是对于不同的优化问题，MATLAB提供了不同的优化属性，可以使用"optimset funname"来查看优化对应的优化情况属性，在后面步骤中将做出演示。

step 4 查看优化属性。在命令窗口中输入"optimset fminimax"，查看该函数的所有相关优化属性如下：

```
>> optimset fminimax
ans =
                  Display: 'final'
              MaxFunEvals: '100*numberofvariables'
                  MaxIter: 400
                   TolFun: 1.0000e-006
                     TolX: 1.0000e-006
              FunValCheck: 'off'
                OutputFcn: []
          ActiveConstrTol: []
      NoStopIfFlatInfeas: []
           BranchStrategy: []
         DerivativeCheck: 'off'
              Diagnostics: 'off'
           DiffMaxChange: 0.1000
           DiffMinChange: 1.0000e-008
      GoalsExactAchieve: []
              GradConstr: 'off'
                  GradObj: 'off'
                  Hessian: 'off'
     ...............................//限于篇幅,这里省略了部分属性列表
        PrecondBandWidth: []
          RelLineSrchBnd: []
  RelLineSrchBndDuration: []
        ShowStatusWindow: []
                  Simplex: []
                   TolCon: 1.0000e-006
                   TolPCG: []
                TolRLPFun: []
              TolXInteger: []
                 TypicalX: 'ones(numberofvariables,1)'
```



在本实例中，也可以通过修改优化的初值条件来重新优化，在这里限于篇幅，就不重复展开详细计算了，请读者自行尝试。

4.5.8 对比实例

例 4.20 使用函数 $f(x)=\frac{1}{1+8x^2}$ 产生的基础数据，然后分别使用"最小最大值优化"和"非线性最小方差优化"的方法对基础数据进行数据拟合，最后绘制拟合图形。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
clear
clc
% 创建各个内联函数
f=inline('1./(1+8*x.*x)','x');
f1=inline('abs(polyval(a,x)-fx)','a','x','fx');
f2=inline('polyval(a,x)-fx','a','x','fx');
% 数据拟合的阶数
N=2;
x0=zeros(1,N+1);
% 产生拟合的数据系列
xx=-2+[ 0:200]'/50;
fx=feval(f,xx);
% 使用最小最大值的方法进行拟合
fm=fminimax(f1,x0,[],[],[],[],[],[],[],[],xx,fx);
% 使用最小方差的方法进行拟合
fln=lsqnonlin(f2,x0,[],[],[],xx,fx);
% 绘制拟合的数据图形
plot(xx,fx,':','LineWidth',1.5)
hold on;
plot(xx,polyval(fm,xx),'m','LineWidth',1.5);
hold on;
plot(xx,polyval(fln,xx),'g','LineWidth',1.5)
grid
axis([-2 2 -0.4 1.1])
legend('Original','MiniMax','LS')
title('The Curve Fitting')
```

step 2 查看程序代码的结果。输入程序代码后，按"Enter"键，得到的图形如图 4.38 所示。

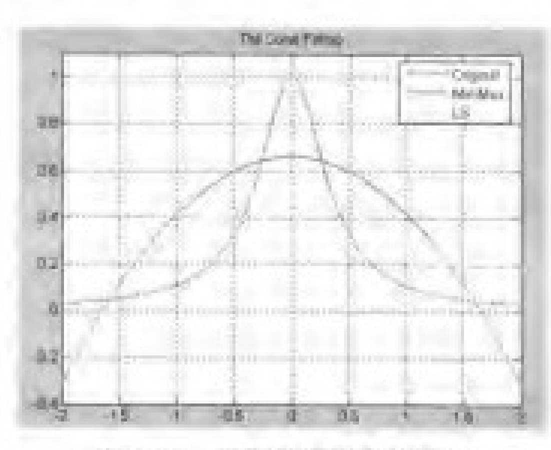


图 4.38 绘制数据拟合的结果

step 3 查看优化求解信息。在显示数据拟合结果的同时，MATLAB 显示关于优化求解的信息：

```
Optimization terminated: Search direction less than 2*options.TolX
```

```
 and maximum constraint violation is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
                                      1
                                     62
                                    101
                                    140
                                    201
Optimization terminated: first-order optimality less than OPTIONS.TolFun,
 and no negative/zero curvature detected in trust region model.
>> fm
fm =
   -0.2424    0.0000    0.6590
>> fln
fln =
   -0.1631   -0.0000    0.4653
```



在上面的程序代码中，fm表示使用最小最大方法求解得到的拟合系数，fln则是使用最小方差的方法求解得到的拟合系数。

4.5.9 线性规划

线性规划是一种特殊的优化问题，目标函数和约束条件都是线性的。对于这种优化问题，可以使用比较特殊的方法来求解。典型的线性规划问题为：

$$\min_x \quad f^T x$$

$$\text{Subject to} \quad A \cdot x \leqslant b$$

$$\text{Aeq} \cdot x = \text{beq}$$

$$lb \leqslant x \leqslant ub$$

在MATLAB中，求解线性规划的命令为linprog，其完整的调用格式如下：

- [x,fval,exitflag,output,lambda] = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)

关于该命令中的各参数的含义，请参考前面的fmincon命令。

例4.21 求解线性规划，其中目标函数为$f(x)=-3x_1-2x_2$，其中参数满足下面的关系式：$0\leqslant x_1,x_2\leqslant 10$。同时，该目标函数满足下面的约束条件：

$$\begin{cases} 2x_1 + x_2 \leqslant 3 \\ 3x_1 + 4x_2 \leqslant 7 \\ -3x_1 + 2x_2 \leqslant 2 \end{cases}$$

为了更加深入地理解线性规划的含义，在本例中将分别使用线性规划和约束优化的方法，求解上面的规划问题。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>> x0=[0 0];
f=[-3 -2];
```

```
A=[ 3 4;2 1] ;
b=[ 7;3] ;
Aeq=[ -3 2] ;
beq=2;
l=[ 0 0] ;
u=[ 10 10] ;
tic
% 使用线性规划的方法来求解优化问题
[ x,fval,exitflag,output,lambda] =linprog(f,A,b,Aeq,beq,l,u);
time_lin=toc;
cons_st=[ A;Aeq] *x-[ b;beq] ;
fcon=inline('-3*x(1)-2*x(2)','x');
[ xc,fvalc,exitflagc,outputc,lambdac] =fmincon(fcon,x0,A,b,Aeq,beq,l,u);
time_con=toc;
```

step 2 查看线性规划求解的结果。在命令窗口中输入下面的程序代码：

```
>> x,fval
x =
     0.3333
     1.5000
fval =
    -4.0000
>> lambda.ineqlin
ans =
     0.6667
     0.0000
>> lambda.eqlin
ans =
    -0.3333
>> lambda.upper
ans =
   1.0e-011 *
     0.2480
     0.2946
>> lambda.lower
ans =
   1.0e-010 *
     0.8574
0.0709
>> cons_st
cons_st =
    -0.0000
    -0.8333
    -0.0000
```



从上面的结果中可以看出，使用线性规划求解的最优解为（0.33，1.5），对应的最优解对应的函数值为-4。同时，第二个约束条件满足，其他的约束条件都没有满足。

step 3 查看约束条件求解的结果。在命令窗口中输入下面的程序代码：

```
>> xc,fvalc
xc =
     0.3333    1.5000
```

```
fvalc =
    -4
>> lambdac.ineqlin
ans =
    0.6667
         0
>> lambdac.eqlin
ans =
   -0.3333
```



从上面求解的结果中可以看出，使用约束条件求解得到的结果和线性规划相同。同时，关于各种约束条件的拉格朗日系数相同。

step 4 对比两种方法的使用时间。在命令窗口中输入下面的程序代码：

```
>> s1 = ['The time of Linear Method is ' num2str(time_lin)];
>> s2 = ['The time of Constrained Method is ' num2str(time_con)];
>>S = strvcat(s1,s2);
>>disp(S)
The time of Linear Method is 0.09
The time of Constrained Method is 0.13
```



从上面的结果中可以看出，使用线性规划的时间为0.09s，而使用约束条件求解的优化问题则需要花费0.13s，性能上明显劣于线性规划的方法。

在本小节的最后，针对该实例绘制出线性规划的图形，通过该图可以直观地理解直线规划的几何含义，如图 4.39 所示。

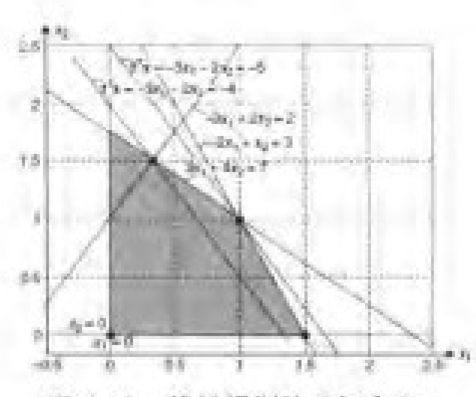


图 4.39　线性规划的几何含义

4.5.10　二次规划

二次规划是指其目标函数最高次数为 2 的优化问题，典型的二次规划的格式如下：

$$\min_{x} \frac{1}{2}x^T H x + f^T x$$

$$\text{Subject to} \quad A \cdot x \leq b$$

$$\text{Aeq} \cdot x = \text{beq}$$
$$lb \leq x \leq ub$$

在 MATLAB 中，求解线性规划的命令为 quadprog，其完整的调用格式如下：

[x,fval,exitflag,output,lambda] = quadprog (f,A,b,Aeq,beq,lb,ub,x0,options)

关于该命令中的各参数的含义，请参考前面的 fmincon 命令。

例 4.22 求解二次规划，其目标函数为 $f(x)=\frac{1}{2}x_1^2+x_2^2-x_1x_2-2x_1-6x_2$，同时，其满足的约束条件为：

$$\begin{cases} x_1+x_2\leqslant 2 \\ -x_1+2x_2\leqslant 2 \\ 2x_1+x_2\leqslant 3 \\ 0\leqslant x_1,x_2 \end{cases}$$

step 1 将二次规划进行转换，转换为标准形式。根据线性代数知识，得到的结果为：

$$H=\begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix},\quad f=\begin{pmatrix} -2 \\ -6 \end{pmatrix},\quad x=\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

step 2 进行二次规划求解。在 MATLAB 的命令窗口中输入下面的代码：

```
%转换成标准形式
>>H = [ 1 -1; -1 2] ;
>>f = [ -2; -6] ;
>>A = [ 1 1; -1 2; 2 1] ;
>>b = [ 2; 2; 3] ;
>>lb = zeros(2,1);
>>[ x,fval,exitflag,output,lambda] = quadprog(H,f,A,b,[ ],[ ],lb);
```

step 3 查看二次规划求解的结果。在 MATLAB 的命令窗口中输入代码：

```
>> x
x =
    0.6667
    1.3333
>> fval
fval =
   -8.2222
>> lambda.ineqlin
ans =
    3.1111
    0.4444
         0
```

在上面的二次规划问题中，求得的最优解为 (0.6667，1.3333)，对应的函数数值为 −8.222。同时，对应的拉格朗日系数为 (3.1111，0.4444，0)。

使用遗传算法求解二次规划

在本节的最后，将介绍如何使用遗传算法来求解这个二次规划问题。如果希望使用本节中的方法，请首先安装 "Genetic Algorithm and Direct Search Toolbox"，遗传算法是近年来发展迅速的计算方法，将在后面章节中详细介绍。下面分步骤介绍如何使用该方法来求解上面的实例。

例 4.23 使用遗传算法来求解上面的二次规划。

step 1 选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function y = quadopt(x);
x = x';
H = [ 1 -1; -1 2];
f = [ -2; -6];
y = 0.5*x'*H*x + f'*x;
```

然后将上面的程序代码保存为 "quadopt.m" 文件，该文件是该二次规划的目标函数。

step 2 设置二次规划求解的条件。在 MATLAB 的命令窗口中输入下面的代码：

```
>> A = [ 1 1; -1 2; 2 1];
>> b = [ 2; 2; 3];
>> lb = zeros(2,1);
>> x0=[ 0,0];
```

step 3 设置规划求解的参数。在命令窗口中输入 "psearchtool"，打开 "Pattern Search Tool" 对话框，并在其中设置规划的参数，如图 4.40 所示。



图 4.40 设置优化条件

在上面的对话框中，分别设置了二次规划的参数：

- 在 "Objective function" 选框中输入 "@quadopt"，其中 quadopt 就是在前面步骤中设置的二次规划的目标函数；
- 在 "Start point" 选框中输入 "x0"，表示进行优化求解的初始条件数值。在前面使用 quadprog 求解时，并没有设置该选项，但是使用遗传算法时必须制定初始条件的数值；
- 在 "Constraints" 面板中设置了二次规划的约束条件。其中 "Linear inequalities" 选框表示不等式约束条件，分别在 "A" 和 "b" 选框中输入对应的参数；"Linear equalities" 选框中表示等式约束条件，在本实例中没有任何等式约束，因此两个选框为空；"Bounds" 选框表示变量的取值范围 $lb \le x \le ub$，在本实例中，只有下限，没有上限，因此在 "Lower" 选框中输入 "lb"。

Step 4 运行规划求解。单击 "Pattern Search Tool" 对话框中的 "Start" 按钮，进行规划求解，并得出对应的结果，如图 4.41 所示。



图 4.41 规划求解的结果

在 "Pattern Search Tool" 对话框的 "Status and results" 选框中，描述该二次规划的结果和求解信息，具体信息如下

```
Pattern search running.
Pattern search terminated.
Objective function value: -8.222220526797175
Optimization terminated: current mesh size 9.5367e-007 is less than
'TolMesh'.
```



说明 从上面的信息可以看出，目标函数的最优解为 -8.222220526797175，最后，还描述了优化求解中止的类型。

在 "Final point" 选框中显示的是该二次规划的最优解坐标 [0.6667, 1.3333]，和前面步骤中使用 quadprog 方法求解得到的最优解相同。

4.6 使用遗传算法求解优化

所谓遗传算法（genetic algorithm）是指，基于自然选择的方法来求解优化问题。由于自然选择是生物进化的重要步骤，因此该方法被称为遗传算法。在遗传算法中，将重复修改个体的总数，在每个计算步骤中，将在当前总数的条件下随机选择个体，作为计算下一代的 "父体"，然后使用这些 "父体" 来产生下一代的 "子体"。在多次连续的数据迭代中，将会通过优化解决方法来实现进化。

在MATLAB中，可以使用遗传算法来解决标准优化算法无法解决或者很难解决的优化问题，例如，当优化问题的目标函数是离散的、不可微的、随机的或者高度非线性化的等，使用遗传算法就会比

前面介绍的优化方法更有效、更方便。

在本节中，将以一个例子来说明如何使用“Genetic Algorithm and Direct Search”工具箱中提供的磣数来进行优化求解。



在“Genetic Algorithm and Direct Search”工具箱中提供大量的函数，方便进行优化求解。如果希望了解这些函数代码，可以在命令窗口中输入“type funname”来查看具体代码。

4.6.1 分析目标函数

例 4.24 使用遗传算法来求解优化问题，求解命令都将使用到“Genetic Algorithm and Direct Search”工具箱中的内置函数。由于这些函数一般都比较复杂，因此，在对应的地方都将尽量作出必要的解释，帮助读者理解优化求解的方法。

step 1 选择命令窗口编辑栏中的“File”⇨“New”⇨“M-File”命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function f = shufcn(y)
for j = 1: size(y,1)
    f(j) = 0.0;
    x = y(j,:);
    temp1 = 0;
    temp2 = 0;
    x1 = x(1);
    x2 = x(2);
    for i = 1:5
        temp1 = temp1 + i.*cos((i+1).*x1+i);
        temp2 = temp2 + i.*cos((i+1).*x2+i);
    end
    f(j) = temp1.*temp2;
end
```

上面的程序代码是在本实例中优化的目标函数，将该代码保存为“shufcn.m”文件。

step 2 绘制上面代码文件的函数图形。在 MATLAB 的命令窗口中输入下面的代码：

```
>>plotobjective(@shufcn,[-2 2; -2 2]);
```

输入上面的代码后，按“Enter”键，得到的图形如图 4.42 所示。

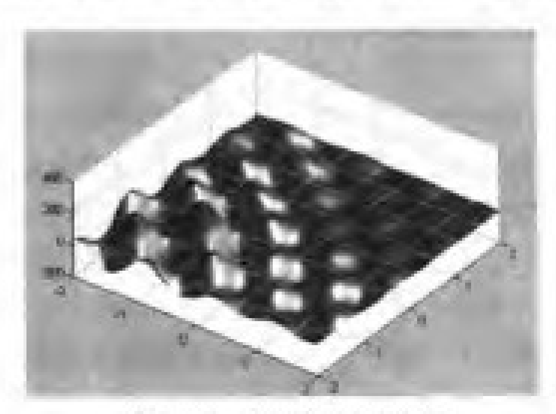


图 4.42　绘制的函数图形

step 3 查看plotobjective函数的M文件程序代码。在上面的命令行中，调用了plotobjective函数，这个函数是"Genetic Algorithm and Direct Search"工具箱中的内置函数，用来绘制目标函数的图形，其具体的程序代码如下：

```
function plotobjective(fcn,range)
%PLOTOBJECTIVE plot a fitness function in two dimensions
%   plotObjective(fcn,range) where range is a 2 by 2 matrix in which
each
%   row holds the min and max values to range over in that dimension.

if(nargin == 0)
    fcn = @rastriginsfcn;
    range = [-5,5;-5,5];
end
pts = 100;
span = diff(range')/(pts - 1);
x = range(1,1): span(1) : range(1,2);
y = range(2,1): span(2) : range(2,2);
pop = zeros(pts * pts,2);

k = 1;
for i = 1:pts
    for j = 1:pts
        pop(k,:) = [x(i),y(j)];
        k = k + 1;
    end
end
values = feval(fcn,pop);
values = reshape(values,pts,pts);

surf(x,y,values)
shading interp
light
lighting phong
hold on
contour(x,y,values)
rotate3d
view(37,60)
```

上面的程序代码多次使用了绘制图形的常见命令，这里大致介绍上面代码的主要含义：首先根据输入的变量情况绘制曲面图，然后绘制相应的等高线。关于具体每个语句的含义，请查看本书中介绍图形绘制的章节。



从上面的绘图结果可以看出，在本小节中之所以选择该函数，是因为该函数本身有很多的数据"浅点"，使用普通的优化方法获得最优解会十分困难。

4.6.2 优化求解

延续上面小节的步骤。

step 1 设置遗传算法的参数。在MATLAB的命令窗口中输入下面的代码：

```
>> FitnessFunction = @shufcn;
numberOfVariables = 2;
```



在使用遗传算法的时候，需要首先设定两个必要的参数：拟合函数（目标函数）和优化问题的变量个数。在本实例中，目标函数为shufcn，变量个数为2。

step 2 运行遗传算法，进行优化求解。在MATLAB的命令窗口中输入下面的代码：

```
>>[ x,Fval,exitFlag,Output] = ga(FitnessFunction,numberOfVariables);
```



在上面的程序代码中，ga是"Genetic Algorithm and Direct Search"工具箱中的内置函数，表示使用遗传算法来求解FitnessFunction的最小值，并返回最优解的各种信息。

step 3 分析ga命令的调用格式。在MATLAB中，ga命令的完整调用格式如下：

```
[ x, fval, reason, output, population, scores] = ga(fitnessfun, nvars,
options)
```

上面命令格式中，各参数的具体含义如下。

- **输入参数**：fitnessfun表示最小值优化问题的目标函数，nvars表示该目标函数中变量的个数，options则是该优化问题的优化属性。
- **输出参数**：变量x表示求解的最优解，变量fval表示的是最优解所在的函数值；变量reqson表示GA算法停止的类型；变量Output则表示解算器运行的各信息，可以查看Output中各分量变量的信息；变量population返回最优解的总体；变量scores则返回最优解总体的总数信息。



在上面命令的输入参数中，options的功能用来设定遗传算法的优化求解属性，和前面章节中介绍的其他优化命令类似，可以使用专门的命令来设置options的具体属性。在Genetic Algorithm and Direct Search Toolbox中，提供gaoptimset命令来设置属性，并返回一个结构体变量options。

step 4 查看优化求解的结果。在MATLAB的命令窗口中输入下面的代码：

```
>> fprintf('The best variable value found was : %g,%g\n', x(1),x(2));
>>fprintf('The best function value found was : %g\n', Fval);
>>fprintf('The number of generations was : %d\n', Output.generations);
>>fprintf('The number of function evaluations was : %d\n', Output.
funccount);
The best variable value found was : -1.44823,-0.797518
The best function value found was : -185.48
The number of generations was : 82
The number of function evaluations was : 1640
```

从上面的结果中可以看出，最优解为[-1.44823, -0.797518]，同时最优解处的函数结果数值为-185.48，最后，遗传算法的总体代数为82。



遗传算法通过使用一系列作用在遗传总体种群的操作，进行优化求解。其中，"种群"是指在某个特定空间中的数据点。在默认情况下，初始种群是随机产生的，下一代的种群则是通过当前种群中每个个体的"适应度"来选取获得的。

4.6.3 设置优化求解的可视化属性

延续上面小节的步骤。

step 1 添加优化结果的可视化属性。使用GAOPTIMSET命令添加遗传算法的图形属性选项：GAPLOTBESTF和GAPLOTSTOPPING，在命令窗口中输入下面的程序代码：

```
>>opts = gaoptimset('PlotFcns',{@gaplotbestf,@gaplotstopping});
```



在上面的程序代码中，gaplotbestf和gaplotstopping都是"Genetic Algorithm and Direct Search"工具箱中的内置绘图函数。其中，函数gaplotbestf的功能是绘制当前种群中的最佳和平均个体的数量；函数gaplotstopping的功能则是绘制满足停止标准的百分比图形。

step 2 查看函数gaplotbestf的程序代码。和前面步骤类似，可以通过type命令来查看上面两个函数的程序代码。这里仅仅列出函数gaplotbestf的程序代码如下：

```
function state = gaplotbestf(options,state,flag)
%GAPLOTBESTF Plots the best score and the mean score.
%   STATE = GAPLOTBESTF(OPTIONS,STATE,FLAG) plots the best score as
well
if(strcmp(flag,'init'))
    set(gca,'xlim',[1,options.Generations]);
    xlabel Generation
    ylabel('Fitness value');
end
hold on;
generation = state.Generation;
best = min(state.Score);
plot(generation,best, '.k');
m = mean(state.Score);
plot(generation,m,'.b');
title(['Best: ',num2str(best),' Mean: ',num2str(m)])
hold off;
```

在上面的程序代码中，引用了state的遗传算法各种属性，因此，该函数图形显示在运行遗传算法过程中的各种计算结果属性。



对于函数gaplotstopping，也可以使用type命令来查看。限于篇幅，在这里就不详细列出其代码了，请读者自行分析。

4.6.4 设置遗传算法的属性

延续上面小节的步骤。

step 1 重新运行遗传算法，进行优化求解。在命令窗口中输入下面的程序代码：

```
>> [x,Fval,exitFlag,Output] = ga(FitnessFunction,numberOfVariables,opts);
>> fprintf('The best variable value found was : %g,%g\n', x(1),x(2));
>>fprintf('The best function value found was : %g\n', Fval);
>>fprintf('The number of generations was : %d\n', Output.generations);
>>fprintf('The number of function evaluations was : %d\n', Output.
funccount);
The best variable value found was : -0.799506,-1.42193
The best function value found was : -186.706
The number of generations was : 30
The number of function evaluations was : 600
```

step 2 查看函数求解过程的图形。除了显示上面的优化结果之外，MATLAB 还会显示对应的函数图形，其中，在运行之中的函数图形如图 4.43 所示。

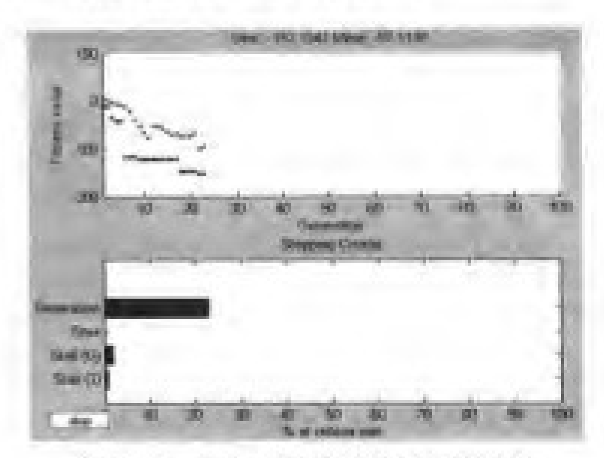


图 4.43 运行中的遗传算法属性图形

step 3 查看函数求解结束的图形。当系统结束遗传算法优化的时候，该函数图形如图 4.44 所示。

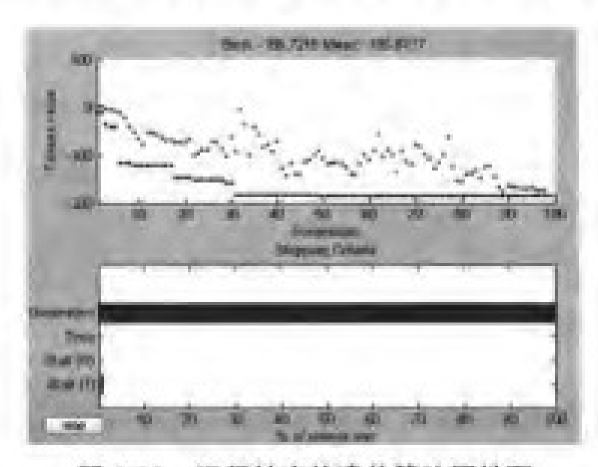


图 4.44 运行结束的遗传算法属性图



从上面的图形中可以看出，该优化求解问题中止的原因在于"Generation"的总数满足系统设置的约束条件，也就是约束满足率为"100%"，其他的约束条件则都没有满足。

4.6.5 设置遗传算法的“种群”属性

延续上面小节的步骤。

step 1 设置遗传算法的“种群”属性。在MATLAB的命令窗口中输入下面的程序代码：

```
>>opts = gaoptimset(opts,'PopInitRange',[-1 0;1 2]);
```

在上面的程序代码中，使用gaoptimset的调用格式：

```
options = gaoptimset(oldopts,'param1',value1,...)
```

在这种调用格式中，引用了原来的优化属性变量oldopts，然后将其中的param1修改为value1，其他的属性数值则保存不变。在上面的步骤中，将默认的初始种群范围修改为[-1 0;1 2]，其他的属性也延用上面步骤中的属性设置。

在遗传算法中，初始种群的范围必须包含两行数据的矩阵，如果该矩阵只有一列数据，则每个变量的变化范围都是相同的。例如，设置“种群”的取值范围为[-1; 1]，则变量的所有初始范围都是从-1到1。



根据前面介绍步骤的遗传算法，种群的属性将直接影响遗传算法的性能，甚至将直接影响计算结果。因此，可以根据目标函数的特征来设置“种群”属性。除了上面步骤中设置的数值范围属性，还可以设置“种群”的大小属性，具体的设置方法请查看对应的帮助文件。

step 2 重新运行遗传算法，求解优化。在MATLAB的命令窗口中输入下面的程序代码：

```
[x,Fval,exitFlag,Output] = ga(FitnessFunction,numberOfVariables,opts);
fprintf('The best variable value found was : %g,%g\n', x(1),x(2));
fprintf('The best function value found was : %g\n', Fval);
fprintf('The number of generations was : %d\n', Output.generations);
fprintf('The number of function evaluations was : %d\n', Output.
funccount);
Optimization terminated: stall generations limit exceeded.
The best variable value found was : -0.790095,4.85739
The best function value found was : -185.551
The number of generations was : 60
The number of function evaluations was : 600
```

step 3 查看函数求解过程的图形。在显示求解信息的同时，MATLAB还会显示对应的优化图形，如图4.45所示。

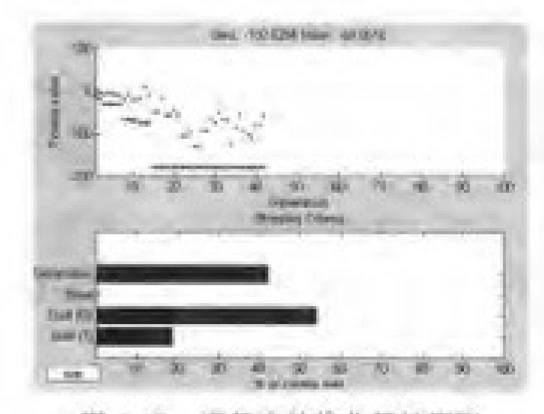


图4.45 运行中的优化属性图形

step 4 查看函数求解结束的图形。当优化求解过程完成后，得到的优化属性的图形如图4.46所示。

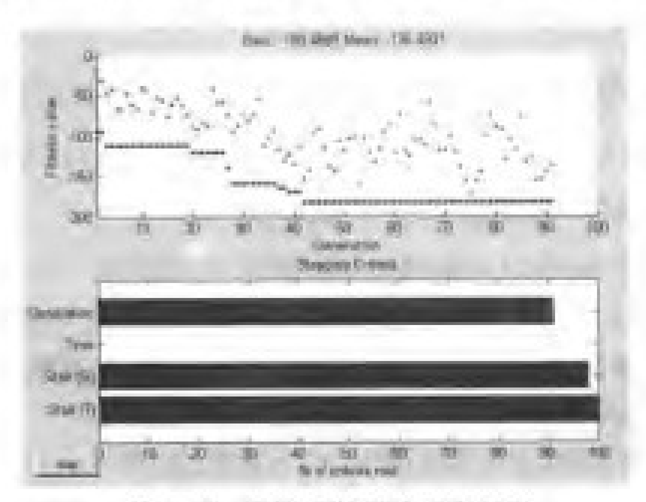


图 4.46　运行后的优化属性图形



从上面的图形中可以看出，该优化求解问题中止的原因在于“Stall”的总数满足系统设置的约束条件，也就是约束满足率为“100%”，其他的约束条件则都没有满足。

4.6.6 设置遗传算法的“中止”属性

延续上面小节的步骤。

step 1 设置遗传算法的“中止”属性。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>>opts = gaoptimset(opts,'Generations',250,'StallGenLimit', 50);
```



在上面的步骤中，将遗传算法中的“中止”最大“代数”设置为250，延迟最大代数为50，在后面的步骤中可以查看该修改的属性对遗传算法的影响。

step 2 重新运行遗传算法，求解优化。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> [x,Fval,exitFlag,Output] = ga(FitnessFunction,numberOfVariables,opts);
>>fprintf('The best variable value found was : %g,%g\n', x(1),x(2));
>>fprintf('The best function value found was : %g\n', Fval);
>>fprintf('The number of generations was : %d\n', Output.generations);
>>fprintf('The number of function evaluations was : %d\n', Output.
funccount);
Optimization terminated: stall generations limit exceeded.
The best variable value found was : -0.818726,-1.51072
The best function value found was : -169.704
The number of generations was : 97
The number of function evaluations was : 1940
```

step 3 查看函数求解结束的图形。在修改了优化属性后，对应的优化属性图形如图 4.47 所示。

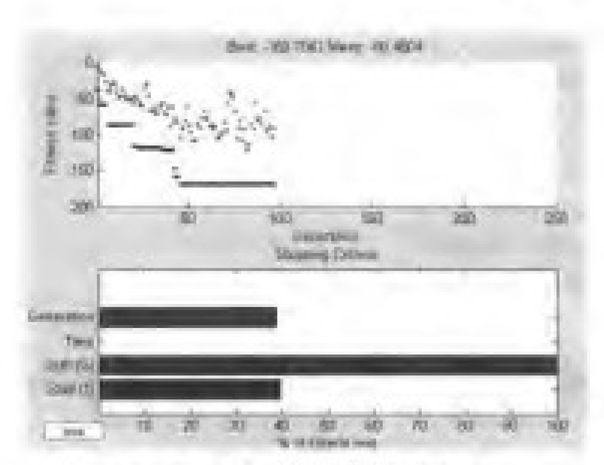


图 4.47 修改属性后的优化图形



在"Genetic Algorithm and Direct Search"工具箱中，还提供其他多个函数和工具，用来分析优化问题，限于篇幅这里就不详细介绍了，请自行查看帮助文件。

4.7 优化求解综合实例：优化"Banana"函数

在优化领域中，二元函数 $f(x,y)=100(y-x^2)^2+(1-x)^2$ 经常作为检测优化方法的对象函数，这个函数又被称为Rosenbrock's Banana测试函数。这个测试函数有一片浅谷，优化算法都很难超越这个浅谷，无法取得最优解。本节将分别使用前面介绍的多种优化方法来求解最优解，并比较各种方法的不同。

由于涉及的步骤比较复杂，下面分小节详细介绍其相应的步骤。

4.7.1 分析目标函数

例 4.25 使用不同的方法求解Banana函数的最小值，并比较各种方法的优劣。

step 1 绘制Banana函数的图形。在MATLAB的命令窗口中输入下面的程序代码：

```
>> figure;
>> x=-2:.2:2;
>> y=-1:.2:3;
>> [xx,yy]=meshgrid(x,y);
>> zz=100*(yy-xx.^2).^2+(1-xx).^2;
>> surfHndl=surface(x,y,zz,'EdgeColor',[.8 .8 .8]);
>> view(10,55);
>> colormap(hsv);
>> hold on;
>> [c,contHndl]=contour3(x,y,zz+50,[100 500],'k');
>> set(contHndl,'Color',[.8 .8 .8]);
>> drawnow
>> plot3(-1.9,2,267.62,'ko', ...
       'MarkerSize',15, ...
       'LineWidth',2, ...
       'EraseMode','none');
>> text(-1.9,2.2,267.62,'   Begin', ...
       'Color',[0 0 0], ...
       'EraseMode','none');
```

```
>> plot3(1,1,0,'ko', ...
      'MarkerSize',15, ...
      'LineWidth',2, ...
      'EraseMode','none');
>> text(0.8,1.4,0,'   End', ...
      'Color',[0 0 0], ...
      'EraseMode','none');
>> title('Banana Function')
>> grid on
```

step 2 查看Banana函数的图形。在输入上面的程序代码后，按"Enter"键，得到的图形如图4.48所示。

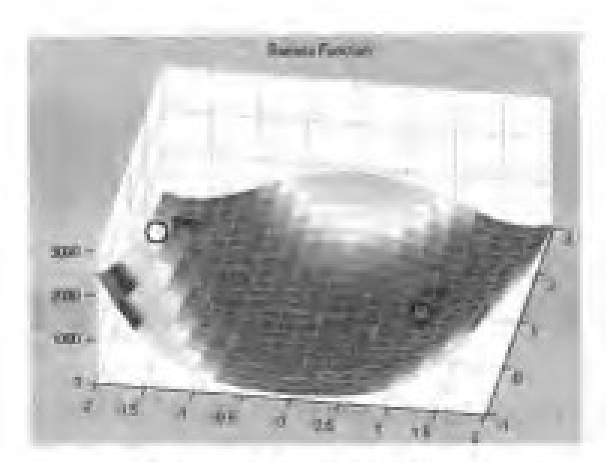


图4.48　绘制目标函数的图形



说明 在上面的图形中，首先绘制了目标函数的曲面图，然后选择（−1.9,2）为起始点（Begin），选择（1,1）为终止点（End）。在上面的数据点中，（−1.9,2）为后面步骤中使用优化求解的初始条件，而（1,1）则是该函数的优化解。因此，比较各种优化方法从起始点到终止点的过程，就可以比较各种优化方法的优劣。

step 3 编写优化方法的输出函数。选择命令窗口编辑栏中的"File"⇨"New"⇨"M-File"命令，打开M文件编辑器，在其中输入下面的程序代码：

```
function stop = bandemOutFcn(x,optimvalues,state,userdata,varargin)
%
% Output function that plots the iterates
stop = false;
if strcmp(state,'iter')
  xpbanplt(x);
end
%xpbanplt函数的程序代码
function out = xpbanplt(currPos,prevPos)
%XPBANPLT Plots one step of solution path.
if nargin==1,
    x1=currPos(1);
    y1=currPos(2);
    z1=100*(y1-x1.^2).^2+(1-x1).^2;
    plot3(x1,y1,z1,'g.', ...
        'EraseMode','none', ...
```

```
        'MarkerSize',25);
    drawnow; % Draws current graph now
    out = [];
elseif nargin==2,
    x1=prevPos(1);
    x2=currPos(1);
    y1=prevPos(2);
    y2=currPos(2);
    z1=100*(y1-x1.^2).^2+(1-x1).^2;
    z2=100*(y2-x2.^2).^2+(1-x2).^2;
    plot3([x1 x2],[y1 y2],[z1 z2],'b-', ...
        'EraseMode','none', ...
        'LineWidth',2);
    plot3([x1 x2],[y1 y2],[z1 z2],'g.', ...
        'EraseMode','none', ...
        'MarkerSize',25);
    drawnow; % Draws current graph now
    out = [];
end
```



输入上面的程序代码后，将代码保存为"bandemOutFcn.m"文件。该程序代码的主要功能在于绘制优化求解迭代的数据点，因此可以显示各优化方法的过程。

4.7.2 "Broyden-Fletcher-Goldfarb-Shanno"优化求解

延续上面小节的步骤。

step 1 使用"Broyden-Fletcher-Goldfarb-Shanno"优化方法求解优化。在MATLAB的命令窗口中输入下面的程序代码：

```
>>x0=[-1.9 2];
>>OPTIONS=optimset('LargeScale','off','OutputFcn',@bandemOutFcn);
>>GRAD='[100*(4*x(1)^3-4*x(1)*x(2))+2*x(1)-2; 100*(2*x(2)-2*x(1)^2)]';
>> f='100*(x(2)-x(1)^2)^2+(1-x(1))^2';
>> OPTIONS = optimset(OPTIONS,'gradobj','on','MaxFunEvals',200, ...
        'InitialHessType','scaled-identity');
>>[x,fval,exitflag,output] = fminunc({f,GRAD},x0,OPTIONS)
```

step 2 查看优化结果。在输入上面的程序代码后，按"Enter"键，得到的结果如下：

```
Optimization terminated: relative infinity-norm of gradient less than
options.TolFun.
x =
    0.9998    0.9996
fval =
  3.4588e-008
exitflag =
     1
output =
       iterations: 34
        funcCount: 50
         stepsize: 1
    firstorderopt: 5.6450e-004
```

```
algorithm: 'medium-scale: Quasi-Newton line search'
  message: [1x85 char]tomsshi
```

从上面的程序代码中可以看出，使用上面的优化方法优化迭代的次数为 34，同时函数评估的次数为 50，得到的结果为（0.9998，0.9996），和最后的最优解（1，1）很接近，表示该优化求解结果正确。



在上面的求解过程中，首先使用optimset命令将“LargeScale”的属性设置为“off”，这是因为在后面的步骤中需要设置“InitialHessType”属性，也就是设置初始的quasi-Newton矩阵类型。这种属性只有在Medium-Scale算法中才具有，因此需要将Large-Scale的功能设置为off。

step 3 查看优化求解过程的图形。除了得到优化结果之外，上面的方法还可以得到求解过程，如图4.49所示。

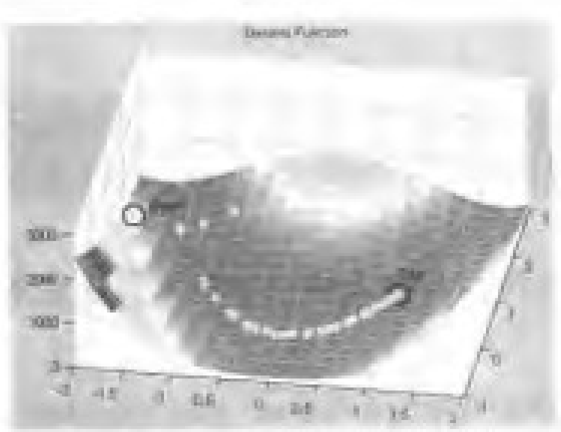


图 4.49　优化求解的过程（一）



在上面的图形中，数据点表示在使用对应的方法进行优化求解的时候，整个过程中迭代的坐标数值。从上面的过程中可以看出，在迭代开始的时候，数据点比较分散，表示开始的时候数据并不收敛。

4.7.3　“Davidon-Fletcher-Powell”优化求解

延续上面小节的步骤。

step 1 使用“Davidon-Fletcher-Powell”优化方法求解优化。在MATLAB的命令窗口中输入下面的程序代码：

```
>>OPTIONS=optimset('LargeScale','off','OutputFcn',@bandemOutFcn);
>>OPTIONS = optimset(OPTIONS,'HessUpdate','dfp','gradobj','on',
'MaxFunEvals', ... 200,'InitialHessType','identity');
>> GRAD='[ 100*(4*x(1)^3-4*x(1)*x(2))+2*x(1)-2; 100*(2*x(2)-2*x(1)^2)]';
>> f='100*(x(2)-x(1)^2)^2+(1-x(1))^2';
>>[ x,fval,exitflag,output] = fminunc({ f,GRAD} ,x0,OPTIONS);
```

step 2 查看程序结果。在输入上面的程序代码后，按“Enter”键，得到的结果如下：

```
Optimization terminated: relative infinity-norm of gradient less than
options.TolFun.
x =
    1.0000    1.0000
fval =
  1.2757e-009
exitflag =
     1
output =
       iterations: 45
        funcCount: 64
         stepsize: 1
    firstorderopt: 0.0011
        algorithm: 'medium-scale: Quasi-Newton line search'
          message: [1x85 char]
```

从上面的程序代码中可以看出，使用上面的优化方法优化迭代的次数为 45，同时函数评估的次数为 64，得到的结果为［1，1］，表示该优化求解结果正确。



"Davidon-Fletcher-Powell" 方法和 "Broyden-Fletcher-Goldfarb-Shanno" 方法最大的区别在于 "InitialHessType" 属性的不同。在 "Broyden-Fletcher-Goldfarb-Shanno" 方法中，"InitialHessType" 属性为 "scaled-identity"，而在 "Davidon-Fletcher-Powell" 方法中，"InitialHessType" 属性为 "identity"。这些参数的具体含义比较复杂，请自行查看对应的帮助文件。

step 3 查看优化求解过程的图形。除了得到优化结果之外，上面的方法还可以得到求解过程，如图 4.50 所示。

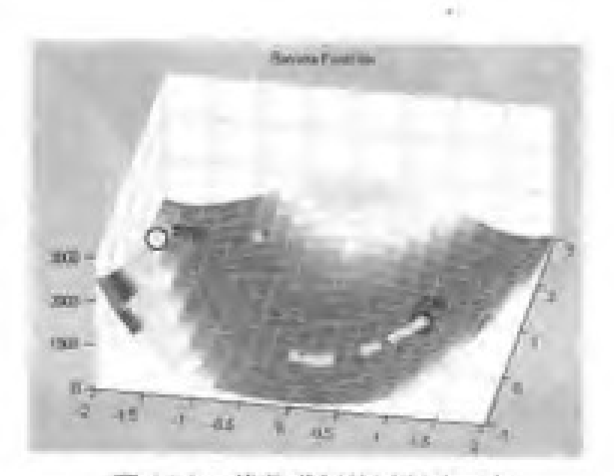


图 4.50　优化求解的过程（二）



从上面的求解过程可以看出，当修改 "InitialHessType" 属性后，优化求解的迭代过程、次数和计算精度将会发生质的飞跃。

4.7.4　"无约束非线性" 优化求解

延续上面小节的步骤。

step 1 使用“无约束非线性”的优化方法求解优化。在MATLAB的命令窗口中输入下面的程序代码:

```
>>OPTIONS = optimset(OPTIONS,'MaxFunEvals',200);
>>f='[ 100*(x(2)-x(1)^2)^2+(1-x(1))^2] ';
>> [ x,fval,exitflag,output] = fminsearch(f,x0,OPTIONS)
```

step 2 查看程序结果。在输入上面的程序代码后，按“Enter”键，得到的结果如下:

```
Exiting: Maximum number of function evaluations has been exceeded
         - increase MaxFunEvals option.
         Current function value: 0.000000
x =
    0.9999    0.9999
fval =
  5.5709e-009
exitflag =
     0
output =
    iterations: 109
     funcCount: 201
     algorithm: 'Nelder-Mead simplex direct search'
       message: [ 1x149 char]
```

step 3 查看优化求解过程的图形。和前面类似，MATLAB显示该优化求解过程，如图4.51所示。

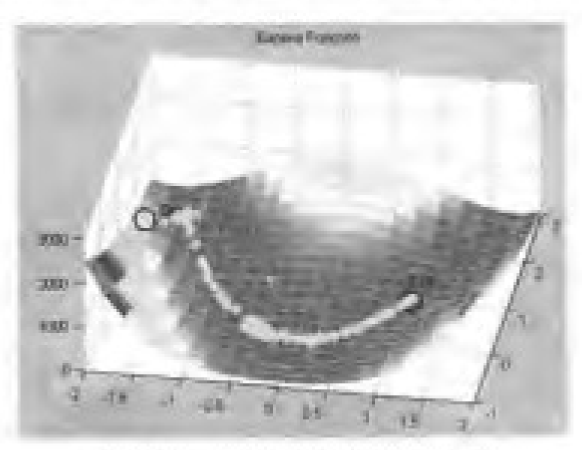


图 4.51　优化求解的过程（三）



从上面的过程可以看出，使用“无约束非线性”的优化方法来求解上面的优化问题时，优化的迭代次数增加，迭代的效率也降低，出现多处反复循环的迭代情况。

4.7.5 “最小方差”优化求解

延续上面小节的步骤。

step 1 使用“最小方差”的优化类型，并且使用Gauss-Newton的优化算法。在MATLAB的命令窗口中输入下面的程序代码:

```
>> OPTIONS = optimset(OPTIONS,'LevenbergMarq','off','MaxFunEvals',200,
```

```
'Jacobian','on');
>> JAC='[-20*x(1), 10; -1, 0]';
>> f='[10*(x(2)-x(1)^2),(1-x(1))]';
>>[x,resnorm,residual,exitflag,output] = lsqnonlin({f,JAC},x0,[],[],
OPTIONS)
```

step 2 查看程序结果。在输入上面的程序代码后，按"Enter"键，得到的结果如下：

```
Optimization terminated: search direction less than TolX.
x =
    1.0000    1.0000
resnorm =
  1.1486e-018
residual =
  1.0e-008 *
    0.1059    0.0165
exitflag =
     4
output =
       iterations: 11
        funcCount: 48
         stepsize: 1.0006
     cgiterations: []
    firstorderopt: []
        algorithm: 'medium-scale: Gauss-Newton, line-search'
          message: 'Optimization terminated: search direction less than
TolX.'
```

step 3 查看优化求解过程的图形。和前面类似，MATLAB 显示该优化求解过程如图 4.52 所示。

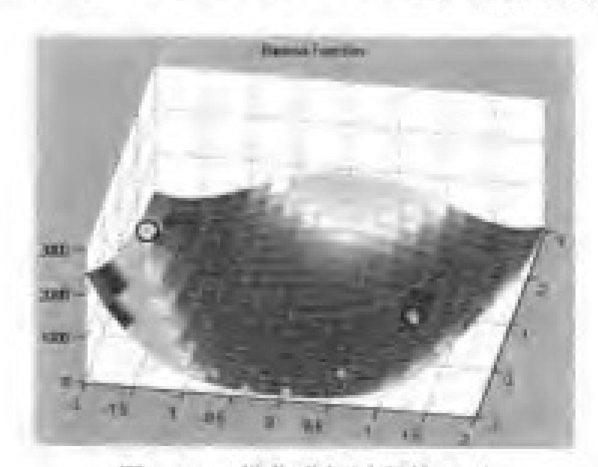


图 4.52 优化求解过程的图形



和前面的优化方法相比，最小方差的求解方法对应的优化迭代次数最小，同时，迭代效率最高，几乎没有任何反复的迭代情况出现。

4.8 优化求解综合实例：复杂的二次规划

在本节中需要求解的规划问题有着明显的实际背景：假定某圆形帐篷的形状是由某个约束条件

的优化问题决定的，可以使用 Optimization Toolbox 中的大型优化功能来解决这个优化问题。假定圆形帐篷需要掩盖方形帐篷柱，具体地讲，帐篷有5个帐篷柱，上面搭着弹性覆盖物。从这个结构中可以看出帐篷的原始形状，这个原始形状对应的就是某能量函数的最小值。能量函数由帐篷表面的位置数据和数据点的梯度的平方模决定。

从上面的介绍可以看出，创建原始帐篷的过程就是求解二次规划的过程，下面分步骤来详细介绍创建的过程。

4.8.1 设置约束条件

例 4.26 使用二次规划的方法来创建原始帐篷的图形。

step 1 绘制原始帐篷的顶柱。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>>largeL = zeros(36);
>>mask = [ 6 7 30 31];
>>largeL(mask,mask) = .3*ones(4);
>>largeL(18:19,18:19) = .5*ones(2);
>>xx = [ 1:5,5:6,6:15,15:16,16:25,25:26,26:30];
>>[ XX,YY] = meshgrid(xx) ;
>>axis([ 1 30 1 30 0 .5],'off');
>>surface(XX,YY,largeL,'facecolor',[ .5 .5 .5],'edgecolor','none');
>>light;
>>colormap(gray);
>>view([ -20,30]);
>>title('The set of tent poles')
```

step 2 查看程序代码的结果。在输入上面的程序代码后，按 "Enter" 键，得到的结果如图 4.53 所示。



图 4.53 绘制帐篷的顶柱



在上面的图形中，largeL 矩阵中的数值表示的是顶柱的高度数值，其中，中间顶柱的高度为 0.5，周边的顶柱高度为 0.3，这些顶柱将主要决定帐篷的主要形状。

step 3 绘制帐篷的下限约束表面。在 MATLAB 的命令窗口中输入下面的程序代码：

```
%  determine a lower bound for the tent
>>L = zeros(30);
>>E = ones(2);
```

```
>>L(15:16,15:16) = .5*E;
>>L(5:6,5:6) = .3*E; L(25:26,5:6) = .3*E;
>>L(5:6,25:26) = .3*E; L(25:26,25:26) = .3*E;
% visualize the constraint
% Add L to the plot.
>>surface(L,'facecolor','none','edgecolor','m');
>>title('Lower Bound Constraint Surface')
```

step 4 查看程序代码的结果。在输入上面的程序代码后，按"Enter"键，得到的结果如图 4.54 所示。

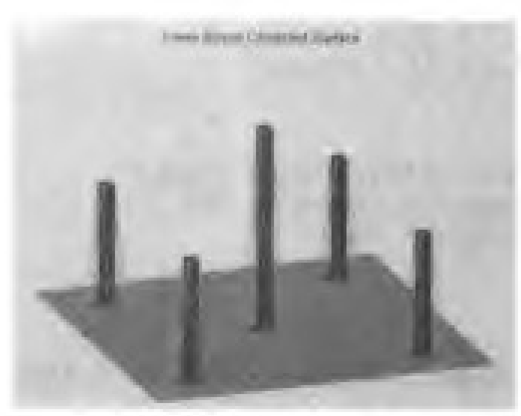


图 4.54　添加下限的曲面

step 5 设置帐篷的优化初始值，并可视化。在命令窗口中输入下面的程序代码：

```
>>sstart = .5*ones(30,30);
% Add it to the plot.
>>surface(sstart,'FaceColor','none','LineStyle','none', ...
   'Marker','.','MarkerEdgeColor','blue')
>>title('Initial Value (blue) and Lower Bound (magenta)');
>>set(gcf,'renderer','zbuffer');  % Markers do not show up in OpenGL.
```

step 6 查看程序代码的结果。在输入上面的程序代码后，按"Enter"键，得到的结果如图 4.55 所示。

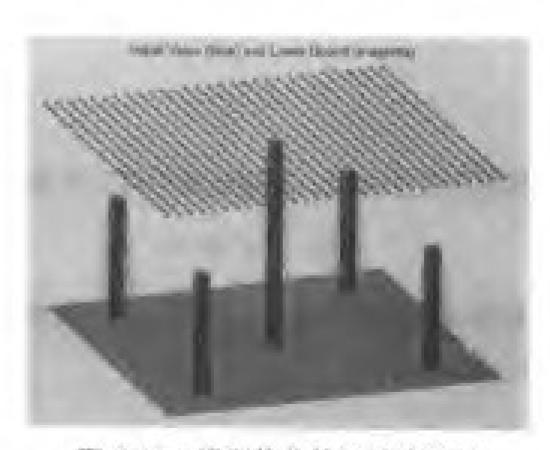


图 4.55　设置优化的初始值表面

step 7 进行向量转换。为了将实际问题转换为标准的优化问题，需要重新将上面的矩阵转换为向量，经过转换后，L 代表初始数值，sstart 表示的是下限。在命令窗口中输入下面的程序代码：

```
>>low = reshape(L,900,1);
>>xstart = reshape(sstart,900,1);
% Illustrate the reordering.
% Draw grid points.
>>xx = 0:4;
>>[ X Y] = meshgrid(xx,xx);
>>gpts = plot(X(:),Y(:),'b.');
>>set(gpts,'markersize',10);
>>axis off; axis([ -2 12 -1.5 5.5] );
>>hold on
>>l(1) = line([ 7.5 6.5] ,[ 2 2.5] );
>>l(2) = line([ 7.5 6.5] ,[ 2 1.5] );
>>l(3) = line([ 7.5 5.5] ,[ 2 2] );
set(l,'color','b');
>>yy = 0.2*xx;
>>zz = [ -1.5+yy,yy,1.5+yy,3+yy,4.5+yy] ;
>>vect = plot(9*ones(25,1),zz,'b.');
>>set(vect,'markersize',9);
>>axis off;
>>hold off;
```

step 8 查看程序代码的结果。在输入上面的程序代码后，按"Enter"键，得到的结果如图4.56所示。



图 4.56　形状转换示例

4.8.2 定义目标函数

延续上面小节的步骤。

step 1 定义目标能量函数。根据前面的介绍，绘制帐篷的目标能量函数为：

$$\min_{x} \ \frac{1}{2}x^T Hx + c^T x$$

其中，$\frac{1}{2}x^T Hx + c^T x$ 是能量函数的离散拟合，同时，其中的变量满足 $lb \leq x$。在MATLAB的命令窗口中输入下面的代码：

```
>>H = delsq(numgrid('S',30+2));
>>h = 1/(30-1);
>>c = -h^2*ones(30^2,1);
```



根据上面的定义，能量函数的每个数据点都只受最邻近的数据点的影响。这个主要特点保证 Hessian 矩阵 ***H*** 是一个稀疏矩阵，结构会比较特殊。同时，矩阵 ***H*** 的这种特性也保证了可以使用 large-scale 算法来求解最优问题。

step 2 查看矩阵结构。在命令窗口中输入下面的代码：

```
spy(H);
title('Structure of Hessian Matrix');
```

step 3 查看矩阵结构图形。在输入上面的程序代码后，按"Enter"键，得到稀疏矩阵的结构，如图 4.57 所示。

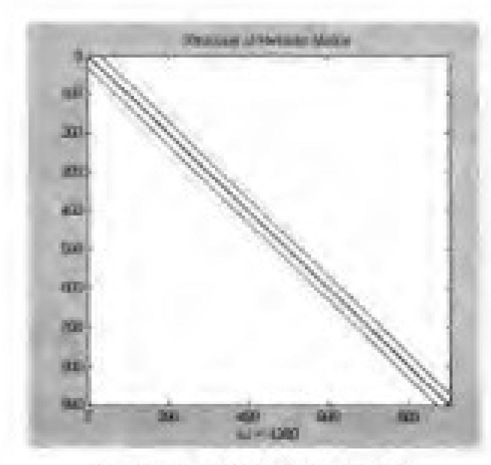


图 4.57　稀疏矩阵的结构

4.8.3 进行优化求解

延续上面小节的步骤。

step 1 绘制数据点的相对位置。由于在求解优化的时候需要显示状态窗口，因此在进行优化运算之前，需要首先以样本数据为例绘制其中的相对位置图形。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>>load tentdata;
>>plot(xXX3,XX3,'b.',xXX1,XX1,'r.',xXX2,XX2,'r.');
>>set(gca,'YTick',[ -1 1]);
>>set(gca,'YTickLabel',{'lower';'upper'});
>>axis([ 1 900 -1 1]);
>>title('Relative position of x(i) to upper and lower bounds (log-scale)')
```

step 2 查看绘制的结果。输入上面代码后，按"Enter"键，得到的图形如图 4.58 所示。



在上面的图形中，每个细节的坐标数值都是相对其上下限而绘制的，如果某个数据点达到其约束条件，则绘制成红色数据点；如果其严格处于上下限之间，则绘制为蓝色数据点。

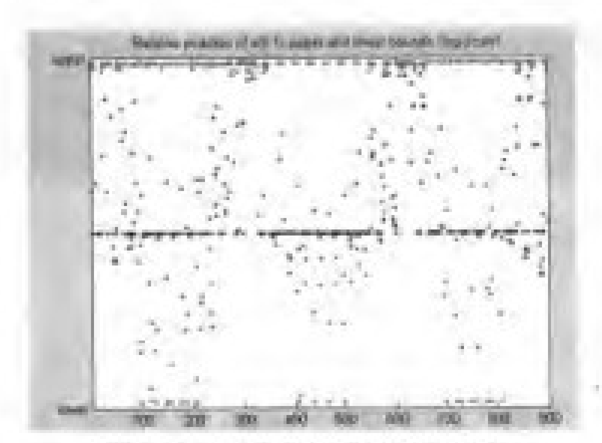


图 4.58 各个数据点的相对坐标

step 3 绘制数据点的归一化的相对梯度。在命令窗口中输入下面的程序代码：

```
>>currplot = plot(xGG2,GG2,'b.',xGG1,GG1,'r.');
>>title('Relative gradient scaled to the range -1 to 1');
>>legend('abs(grad) > tol','abs(grad) <= tol',4);
>>axis([ 1 900 -1 1]);
>>set(gca,'YTick',[-1 0 1]);
>>ylabel('gradient')
```

step 4 查看绘制结果。在输入上面代码后，按"Enter"键，得到的图形如图 4.59 所示。

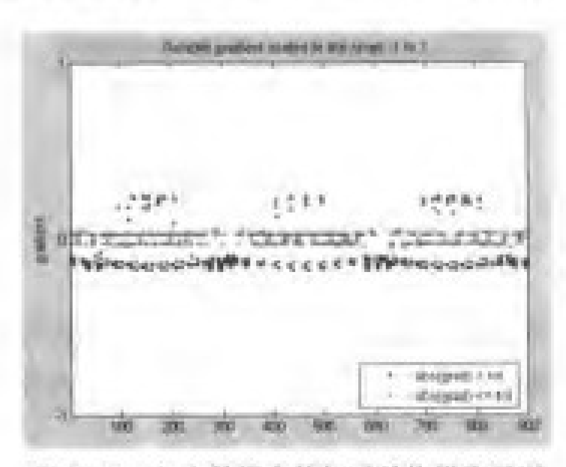


图 4.59 各个数据点的相对单位梯度图形



在上面的图形中，当某组件的梯度在某个限度内接近0，则绘制为红色；其他的数据点则绘制成蓝色。最后，在上面的图形中，tentdata是示例数据，和本规划无关。

step 5 设置优化属性，进行优化求解。在命令窗口中输入下面的程序代码：

```
>>options = optimset('LargeScale','on','display','off', ...
   'ShowStatusWindow','iterplus');
>>x = quadprog(H, c, [], [], [], [], low, [], xstart, options);
```

step 6 查看优化求解过程的图形。在上面的程序代码中，首先设置需要显示关于迭代过程的状态窗口，然后在该优化条件下，进行二次规划求解，得到的"Progress Information"对话框如

图 4.60 所示。

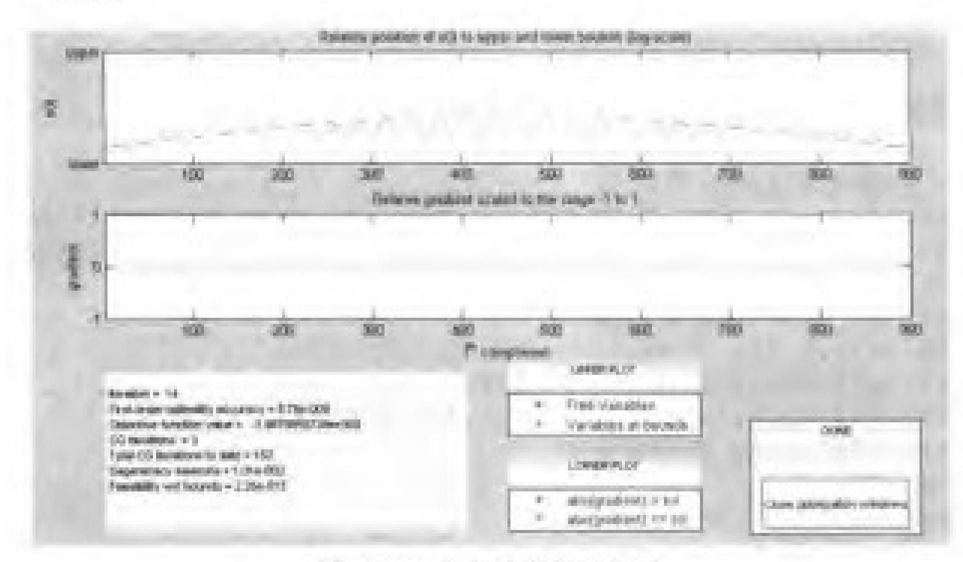


图 4.60　显示迭代过程信息



上面的对话框是优化过程结束后的迭代信息，上面一个图形表示各数据的相对位置，下面一个图形表示各数据点的相对梯度。同时，在对话框的下方左侧显示关于迭代过程的主要信息。

step 7 查看优化信息的图形。同时，运行优化求解后，MATLAB 还会显示 "Algorithm Performance Statistics" 对话框。该对话框显示关于优化算法运行的各种信息，如图 4.61 所示。

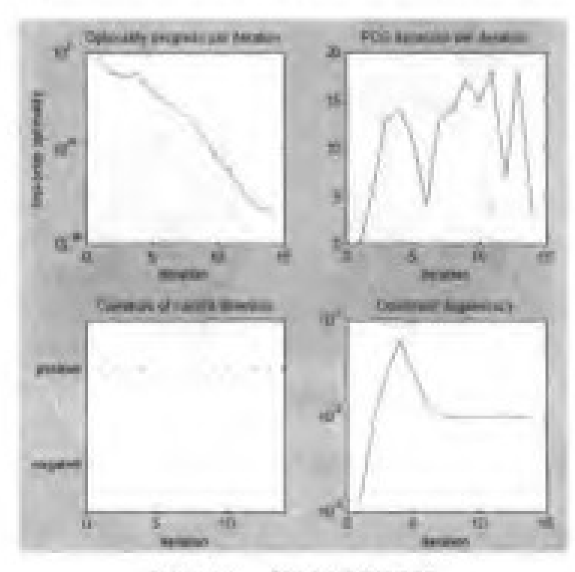


图 4.61　算法运行属性



在上面的图形中，主要显示在优化过程中每次迭代运行的优化回归属性，具体的属性比较复杂，在这里就不详细介绍了。

4.8.4 绘制优化求解的结果

延续上面小节的步骤。

step 1 利用优化结果绘制曲面图。在命令窗口中输入下面的程序代码：

```
>>S = reshape(x,30,30);
% Close figures that QUADPROG creates (if they are still open).
>>delete(findobj(0,'Name','Algorithm Performance Statistics'))
>>delete(findobj(0,'Name','Progress Information'))
% Plot the starting surface.
>>subplot(1,2,1);
>>surf(L,'facecolor',[.5 .5 .5]);
>>surface(sstart,'edgecolor','b','facecolor','none');
>>title('Starting Surface')
>>axis off
>>axis tight;
>>view([-20,30]);
>>subplot(1,2,2);
>>surf(L,'facecolor',[.5 .5 .5]);
>>surface(S,'edgecolor','b','facecolor','none');
>>title('Solution Surface')
>>axis off
>>axis tight;
>>view([-20,30]);
```

step 2 查看程序代码的结果。在输入上面代码后，按"Enter"键，得到的图形如图 4.62 所示。

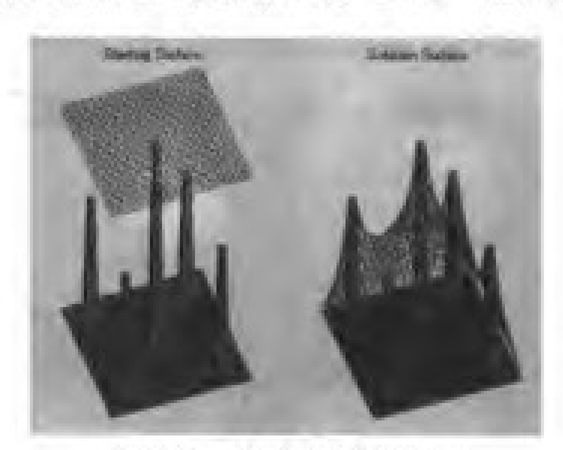


图 4.62　绘制求解的曲面图

step 3 设置帐篷表面的属性。在命令窗口中输入下面的程序代码：

```
>>subplot(1,1,1)
>>surf(L,'facecolor',[0 0 0]);
>>hold on;
>>surfl(S);
>>hold off;
>>axis tight;
>>axis off;
>>view([-20,30]);
```

step 4 查看代码的结果。输入上面代码后，按"Enter"键，得到的图形如图 4.63 所示。

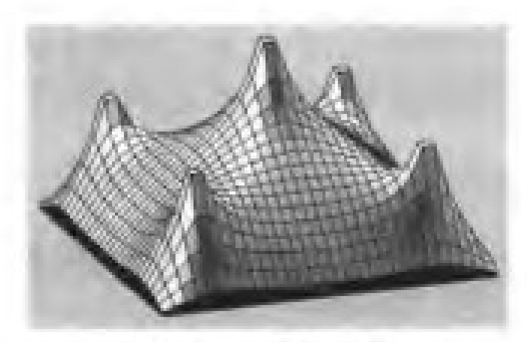


图 4.63 绘制的帐篷

4.9 常微分方程

常微分方程在工程和科学领域有着十分广泛的应用，许多工程的模块都可以抽象成为常微分方程，但是限于当前的数学知识，大部分的常微分方程都没有解析解，或者求取的解析解不方便后续的分析。因此，如何运用数值方法求解常微分方程将是十分重要的内容。

在MATLAB中，提供多种数值求解常微分方程的命令，来求解多种关于常微分方程的问题：初值问题、延迟常微分方程和微分的边界问题。本节将分别使用具体的例子来介绍这些问题的求解方法。

4.9.1 显性常微分方程

在数学理论中，显性常微分方程具有下面的形式：

$$y'=f(t,y) \quad y(t_0)=y_0$$

在上面的公式中，y'、y和y_0都是以变量为参数的向量，其中$y(t_0)=y_0$就是该常微分方程的初值条件，微分方程需要求解的是$y(t)$。

尽管显性常微分方程在形式上具有统一性，但是在微分方程的性质上有很大的区别，因此MATLAB对于求解常微分方程提供ode45、ode23、ode113、ode15s、ode23s、ode23t和ode23tb等多个命令，表4.2列出了各命令的使用情况。

表 4.2 常微分方程组的解法

方法	采用的算法	精度	说明
ode45	四阶/五阶 龙格－库塔	中等	属于单步算法，不需要附加初始值；改变步长也不改变结果
ode23	二阶/三阶 龙格－库塔	低	属于单步算法，在误差容许范围比较宽时性能比ode45好
ode113	可变阶Adams PECE算法	低－高	属于多步算法，适合解决误差容许范围比较严格的情况
ode15s	可变阶数值微分算法	低－中	属于多步算法，适合解决刚性系统
ode23s	基于改进的Rosen公式	低	属于单步算法，适合于解决误差容许范围比较宽的时候
ode23t	自由内插实现梯形公式	低	情况使用轻微刚性系统，给出的解没有数值衰减
ode23tb	TR－BDF2方法	低	对于误差容许范围比较宽的情况

对于常微分方程，在求解过程中需要接触到一个十分重要的概念——刚性（stiffness），一个常微分方程组的刚性将直接决定求解方法和精度。如果微分方程的Jocabian矩阵的特征值相差悬殊，则

这个方程组就被称为刚性方程组。对于刚性方程组，为了保持解法的稳定，步长选取会很困难，因此有些方法将无法用来求解刚性方程组，而有些方法由于对方程组的刚性要求不严，则可以用来求解刚性方程组。

具体来讲，MATLAB 中常见的方法对刚性方程组的适用范围如下。

- 刚性方程组：ode15s、ode23s、ode23b 和 ode23t（适合轻微刚性）
- 非刚性方程组：ode45、ode23 和 ode113

在MATLAB中，用来求解显性常微分方程组的各种命令的调用格式是完全一致的。为了叙述方便，下面在介绍解算命令时将用 solver 来代替具体的命令，在实际运用时需要将 solver 改为对应的命令（例如 ode45），具体的调用格式如下：

- [t,Y] = solver(odefun,tspan,y0,options)
- [t,Y,TE,YE,IE] = solver(odefun,tspan,y0,options)

在上面的调用格式中，各参数的具体含义如下：

- 参数"odefun"表示 ODE 函数的名称；
- 参数"tspan"有两种可能，当 tspan 表示二元向量$[t_0,t_f]$时，tspan 是用来定义求解数值解的时间区间；当 tspan 表示多元向量$[t_0,t_1,\cdots,t_f]$时，命令将会在 tspan 定义的时间序列进行数值求解，此时 tspan 的元素必须按照单调次序排列；
- 参数 y0 则表示微分方程的初始数值；
- 参数 options 用来设置算法的参数，当输入变量的个数为 3 时，算法将会采用默认设置，options 的具体数值可以由函数 odeset 来获得；
- 参数 t 是所求数值解的自变量数据列向量；
- 参数 Y 表示所求微分方程的因变量数据矩阵；
- 参数 TE,YE,IE 只有在设置"Event"记录时，才有对应的输出数据。



关于常微分方程的属性设置 options 和"Event"记录属性的内容，暂将在后面的章节中详细介绍，这里就不展开分析了。

例 4.27 使用 MATLAB 的命令来求解非刚性方程组的数值解：

$$\begin{cases} y_1' = y_2 y_3 \\ y_2' = -y_1 y_3 \\ y_3' = -0.51 y_1 y_2 \end{cases}$$

其中初值条件为：$y_1(0)=0$，$y_2(0)=1$ 和 $y_3(0)=1$。

step 1 选择命令窗口编辑栏中的"File"⇨"New"⇨"M-File"命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function dydt = euler(t,y)
dydt = [     y(2)*y(3)
            -y(1)*y(3)
         -0.51*y(1)*y(2) ];
```

在输入程序代码后，将函数保存为"euler.m"文件，该函数将是微分方程组函数。

step 2 在 MATLAB 的命令窗口中输入下面的命令：

```
% 定义微分方程求解的时间区间
>> tspan = [ 0 12];
% 定义微分方程的初值条件
>>y0 = [ 0; 1; 1];
% 进行微分方程求解
>> [ t,Y] = ode45(@euler,tspan,y0);
% 绘制微分方程组的计算结果
>>plot(t,Y(:,1),'r-','LineWidth',1.5);
>>hold on;
>>plot(t,Y(:,2),'g-.','LineWidth',1.5);
>>hold on;
>>plot(t,Y(:,3),'b.')
% 设置绘制图形的属性
>>axis([ 0 12 -1.2 1.2])
>>legend('Y(1)','Y(2)','Y(3)')
>>grid on
>>title('The Numerical Solution of Euler Equation')
```

step 3 查看程序代码的结果。在输入上面的程序代码后，按"Enter"键，得到的结果如图 4.64 所示。

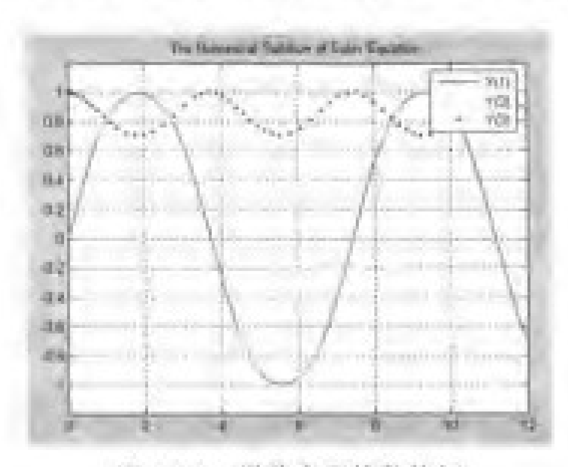


图 4.64　微分方程的数值解

上面的例子比较简单，但是基本上演示了求解常微分方程的步骤：

- 把 t 和 y 作为输入变量，将 y′ 作为输出变量，编写常微分方程的 M 文件。
- 设置微分方程求解的参数：时间跨度、初始数值和求解器参数等；
- 调用合适的微分方程求解命令，求解上面的微分方程，得到数值解。



在 MATLAB 中求解的微分方程得到的都是数值解，而无法得到解析解。为了显示这些数值解的变化情况，经常需要借助 MATLAB 的图形功能。涉及到这些部分的内容，请查看相应的章节。

例 4.28　使用 MATLAB 的命令来求解下面刚性方程组的数值解：

$$
\begin{cases}
y_1' = -0.04y_1 + 10^4 y_2 y_3 \\
y_2' = -0.04y_1 - 10^4 y_2 y_3 - 3\times10^7 y_2^2 \\
y_3' = 3\times10^7 y_2^2
\end{cases}
$$

其中初值条件为：$y_1(0)=0$，$y_2(0)=0$和$y_3(0)=0$。

step 1 选择命令窗口编辑栏中的"File"⇨"New"⇨"M-File"命令，打开M文件编辑器，在其中输入下面的程序代码：

```
function dydt = stiffex1(t,y)
dydt = [  (-0.04*y(1) + 1e4*y(2)*y(3))
          (0.04*y(1) - 1e4*y(2)*y(3) - 3e7*y(2)^2)
          3e7*y(2)^2 ];
```

在输入程序代码后，将函数保存为"stiffex1.m"文件，该函数将是微分方程组函数。

step 2 在MATLAB的命令窗口中输入下面的命令：

```
>> tspan=[0 4*logspace(-6,6)];
>> y0 = [1; 0; 0];
>> [t,y] = ode15s(@stiffex1,tspan,y0);
>> y(:,2) = 1e4*y(:,2);
>> semilogx(t,y(:,1),'r-','LineWidth',1.5);hold on;
>> semilogx(t,y(:,2),'g-.','LineWidth',1.5);hold on;
>> semilogx(t,y(:,3),'b.');
>> ylabel('1e4 * y(:,2)');
>>  axis([10^(-10) 10^10 -0.1 1.1])
>> legend('y1','y2','y3')
>> set(gca,'Ytick',[-0.1:0.1:1.1])
>> title('Stiff Equation solved by ODE15S');
>> grid
```

step 3 查看程序代码的结果。在输入上面的程序代码后，按"Enter"键，得到的结果如图4.65所示。

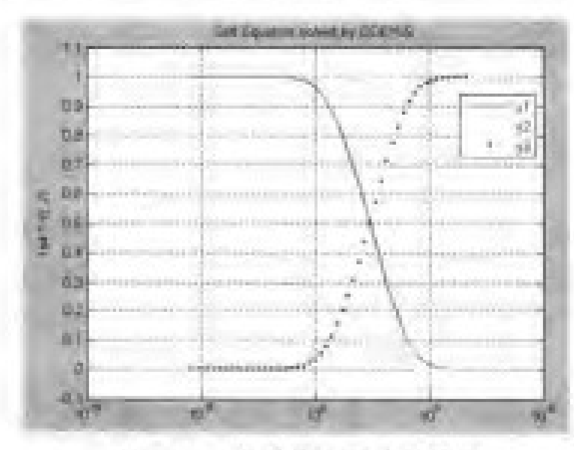


图4.65 常微分方程的数值解



在上面的方程组中，由于其jacobian矩阵的特征值相差比较大，因此属于刚性方程组，需要使用ode15s命令进行求解。可以自行尝试使用ode45命令求解的方法。

4.9.2 设置允许误差属性

前面需要求解的微分方程比较简单，因此在求解过程中并没有设置解法器参数options。如果需要求解的微分方程比较复杂，则需要根据情况来设置相应的参数。在MATLAB中，可以设置各种解法器属性的参数，例如允许误差参数、输出参数、Jacobian 矩阵参数和步长参数等。

在详细介绍各种解法属性之前，首先使用命令来定义各种属性。在MATLAB中，定义解法器参数的命令为odeset函数，其常用的调用格式如下：

- options = odeset('name1',value1,'name2',value2,...)，用参数名称和参数数值来分别定义解法器的各种参数。
- options = odeset(oldopts,'name1',value1,...)，修改原来解法器options结构体oldopts，改变的是指定参数的数值。



在设置解法器的属性参数之前，有必要首先了解各参数的默认数值，可以使用odeset命令来查看所有属性的默认数值。

从下面开始，将详细介绍各种解法器参数的设置工作。其中，允许误差（Error Tolerance）参数如表4.3所示。

表4.3 允许误差的属性参数

属性名称	取值	默认数值	说明
RelTol	正标量	1e-3	用于所有分量的相对误差，解法器的积分估计误差必须小于相对误差和数值解的乘积并且小于相对误差
AbsTol	正标量或者向量	1e-6	绝对误差允许的范围，如果是标量，则该绝对误差用于所有的分量；如果是向量则单独指定每一个分量的绝对误差
NormControl	on或者off	off	如果该属性数值为on，则采用积分估计误差的模来控制精度，如果该属性数值为off，则采用更加严格的精度控制计算



关于微分方程的计算误差理论比较复杂，需要了解的是，对于同一个微分方程的问题，设置不同的误差属性，会直接影响方程的求解时间和效率。

例4.29 设置允许误差的属性，重新求解例4.26中的非刚性微分方程组。

step 1 在MATLAB的命令窗口中输入下面的命令：

```
>> options = odeset('RelTol',1e-4,'AbsTol',[ 1e-4 1e-4 1e-5] );
>>tic;
>> [ t1,Y1] = ode45(@euler,[ 0 12],[ 0 1 1] ,options);
>>time_set=toc;
>>tic;
>> [ t,Y] = ode45(@euler,[ 0 12] ,[ 0 1 1] );
>>time_default=toc;
```

step 2 查看程序代码的结果。在输入上面的程序代码后，可以查看两种属性条件下所消耗的时间：

```
% 设置属性后消耗的时间
>> time_set
time_set =
    0.0100
% 默认属性所使用的时间
>> time_default
time_default =
    0.0300
```



从上面的结果中可以看出，通过设置误差属性后，求解该微分方程所用时间为0.01s，而使用默认的误差属性所用的时间为0.03s。

step 3 比较两种方法的精度。在MATLAB的命令窗口中输入下面的命令：

```
% 设置属性和默认属性条件下的时间间隔
>> size_set=size(t1,1)
size_default=size(t,1)
size_set =
    85
size_default =
    77
```



从上面的结果中可以看出，使用设置误差属性后的时间间隔数为85，使用默认属性的时间间隔数为77，因此，设置误差后的精度比默认情况要高。

step 4 绘制设置解法器属性后的数值解图形。在命令窗口中输入下面的命令：

```
% 定义微分方程求解的时间区间
>> tspan = [ 0 12];
% 定义微分方程的初值条件
>>y0 = [ 0; 1; 1];
% 进行微分方程求解
>> [ t1,Y1] = ode45(@euler,tspan,y0,options);
% 绘制微分方程组的计算结果
>>plot(t1,Y1(:,1),'r-','LineWidth',1.5);
>>hold on;
>>plot(t1,Y1(:,2),'g-.','LineWidth',1.5);
>>hold on;
>>plot(t1,Y1(:,3),'b.')
% 设置绘制图形的属性
>>axis([ 0 12 -1.2 1.2])
>>legend('Y(1)','Y(2)','Y(3)')
>>grid on
>>title('The Numerical Solution of Euler Equation')
```

step 5 查看程序代码的结果。在输入上面的程序代码后，按"Enter"键，得到的图形如图4.66所示。

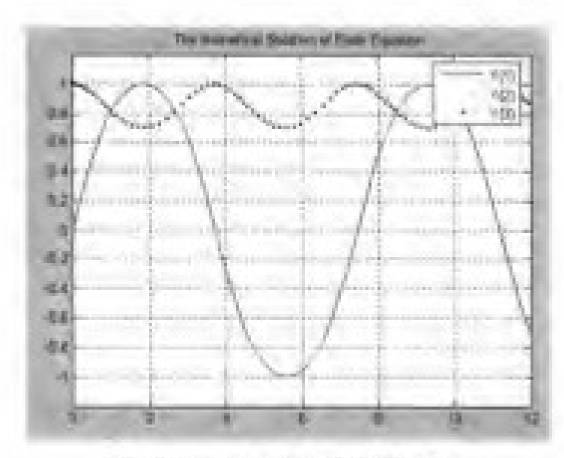


图 4.66 设置属性后的图形



从上面的图形中可以看出，该图形和默认属性条件下求解的图形在外观上没有太大的区别，这是因为精度差别不能在外观上显示比较出来。

4.9.3 设置输出参数属性

在 MATLAB 中，解法器输出（Solver Output）参数如表 4.4 所示。

表 4.4 解法器输出的属性参数

属性名称	取值	默认数值	说明
OutputFcn	函数句柄	@odeplot 或者[]	在每个计算时间步长结束后，使用该函数输出结果。在调用 ode 类的函数时，如果有输出变量，在默认情况下将使用函数 odeplot 绘制结果；如果没有输出变量，则不输出结果
OutputSel	正整数向量	分量的下标	该向量所包含的下标所对应的分量被送到输出函数 OutputFcn 中输出。默认情况下，将输出所有的下标
Refine	正整数	1 或者 4	如果该数值大于 1，则输出结果会被插值
Stats	on 或者 off	off	选中该属性选项，输出计算所消耗的时间



在参数"OutputFcn"中，除了可以使用 odeplot 命令之外，还可以使用 odephas2、odephas3 和 odeprint 等函数命令，也可以自行编写对应的绘制函数。

例 4.30 设置解法器输出的属性，重新求解例 4.26 中的非刚性微分方程组。

step 1 使用默认属性。在 MATLAB 的命令窗口中输入下面的命令：

```
>> tspan = [ 0 12];
y0 = [ 0; 1; 1];
figure;
ode45(@euler,tspan,y0);
```

step 2 查看程序代码的结果。在输入上面的程序代码后，按"Enter"键，得到的结果如图 4.67 所示。

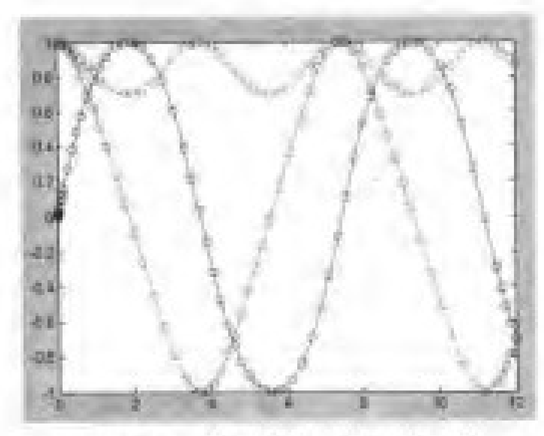


图 4.67　默认输出条件下的图形

step 3 重新设置输出属性，并求解微分方程。在命令窗口中输入下面的命令：

```
>> options=odeset('OutputFcn',@odephas3,'Stats','on');
>> figure;
>> ode45(@euler,tspan,y0,options);
```

step 4 查看程序代码的结果。在输入上面的程序代码后，按"Enter"键，得到的结果如图 4.68 所示。

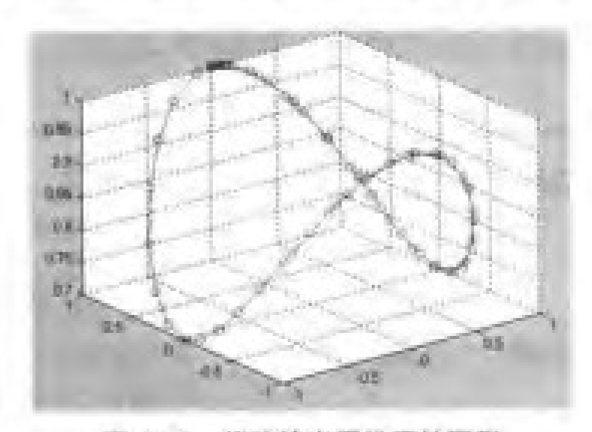


图 4.68　修改输出属性后的图形

同时，在 MATLAB 的命令窗口中显示求解的各种信息：

```
19 successful steps
2 failed attempts
127 function evaluations
```



如果不希望采用系统默认的绘制函数，可以使用两种方法来进行修改：在默认绘图函数的基础上，进行必要的修改，得出自行编写的绘图函数；在使用默认函数绘制的图形上，使用绘图命令进行必要的编辑。

step 5 查看默认函数的程序代码。如果希望使用第一种方法来编写绘图函数，则首先需要分析默认函数的程序代码，以函数"odeplot"为例，其默认的保存路径为...MATLAB7.0\toolbox\matlab\funfun，同时其程序代码如下：

```
function status = odeplot(t,y,flag,varargin)
status = 0;                                % Assume stop button wasn't pushed.
```

```
chunk = 128;                                % Memory is allocated in chunks.
if nargin < 3 || isempty(flag) % odeplot(t,y) [v5 syntax] or odeplot(t,
y,'')
  ud = get(gcf,'UserData');
  % Append t and y to ud.t and ud.y, allocating if necessary.
  nt = length(t);
  chunk = max(chunk,nt);
  [rows,cols] = size(ud.y);
  oldi = ud.i;
  newi = oldi + nt;
  if newi > rows
    ud.t = [ud.t; zeros(chunk,1)];
    ud.y = [ud.y; zeros(chunk,cols)];
  end
  ud.t(oldi+1:newi) = t;
  ud.y(oldi+1:newi,:) = y.';
  ud.i = newi;
  set(gcf,'UserData',ud);
  if ud.stop == 1                           % Has stop button been pushed?
    status = 1;
  else
    ylim = get(gca,'ylim');
    % Replot everything if out of axis range or if just initialized.
    if (oldi == 1) || (min(y(:)) < ylim(1)) || (ylim(2) < max(y(:)))
      for j = 1:cols
        set(ud.lines(j),'Xdata',ud.t(1:newi),'Ydata',ud.y(1:newi,j));
      end
    else
      % Plot only the new data.
      for j = 1:cols
       set(ud.line(j),'Xdata',ud.t(oldi:newi),'Ydata',ud.y(oldi:newi,j));
      end
    end
  end
  else
  switch(flag)
  case 'init'                               % odeplot(tspan,y0,'init')
    ud = [];
    cols = length(y);
    ud.t = zeros(chunk,1);
    ud.y = zeros(chunk,cols);
    ud.i = 1;
    ud.t(1) = t(1);
    ud.y(1,:) = y.';
    f = figure(gcf);
    if ~ishold
      ud.lines = plot(ud.t(1),ud.y(1,:),'-o');
      hold on
      ud.line = plot(ud.t(1),ud.y(1,:),'-o','EraseMode','none');
      hold off
      set(gca,'XLim',[min(t) max(t)]);
    else
      ud.lines = plot(ud.t(1),ud.y(1,:),'-o','EraseMode','none');
      ud.line = plot(ud.t(1),ud.y(1,:),'-o','EraseMode','none');
    end
    % The STOP button.
```

```
    h = findobj(f,'Tag','stop');
    if isempty(h)
      ud.stop = 0;
      pos = get(0,'DefaultUicontrolPosition');
      pos(1) = pos(1) - 15;
      pos(2) = pos(2) - 15;
      str = 'ud=get(gcf,''UserData''); ud.stop=1; set(gcf,''UserData'',ud);';
      uicontrol( ...
           'Style','push', ...
           'String','Stop', ...
           'Position',pos, ...
           'Callback',str, ...
           'Tag','stop');
    else
      set(h,'Visible','on');              % make sure it's visible
      if ishold
        oud = get(f,'UserData');
       ud.stop = oud.stop;              % don't change old ud.stop status
      else
        ud.stop = 0;
      end
    end
    set(f,'UserData',ud);
     case 'done'                          % odeplot([],[],'done')
    f = gcf;
    ud = get(f,'UserData');
    ud.t = ud.t(1:ud.i);
    ud.y = ud.y(1:ud.i,:);
    set(f,'UserData',ud);
    cols = size(ud.y,2);
    for j = 1:cols
      set(ud.lines(j),'Xdata',ud.t,'Ydata',ud.y(:,j));
    end
    if ~ishold
      set(findobj(f,'Tag','stop'),'Visible','off');
      set(gca,'XLimMode','auto');
     refresh;                       % redraw figure to remove marker frags
    end
      end
end
drawnow;
```



可以选择上面程序代码的相应位置，修改图形的相关属性，然后保存函数就可以设置自定义的函数，具体的做法就不介绍了。

4.9.4 设置解法器其他属性

关于解法器的其他属性，这里就不详细介绍了。感兴趣的读者可以查阅MATLAB中的相关帮助文件，下面选择一个比较综合的例子说明如何设置属性，来求解具体的实例。

例 4.31 求解小球反弹的轨迹模型，该小球在反弹后速度变为下落速度的90%，使用微分方程来求解小球运动的轨迹。

step 1 选择命令窗口编辑栏中的“File”⇨“New”⇨“M-File”命令，打开M文件编辑器，在其中输入下面的程序代码：

```
function dydt =bounce (t,y)
dydt = [ y(2); -9.8] ;
```

在输入上面的程序代码后，将该程序代码保存为“bounce.m”文件。

step 1 选择命令窗口编辑栏中的“File”⇨“New”⇨“M-File”命令，打开M文件编辑器，在其中输入下面的程序代码：

```
function [ value,isterminal,direction] = events(t,y)
% Locate the time when height passes through zero in a decreasing direction
% and stop integration.
value = y(1);       % detect height = 0
isterminal = 1;     % stop the integration
direction = -1;     % negative direc
```

输入上面的程序代码后，将该程序代码保存为“events.m”文件，该函数为该微分方程的求解过程中的“定位事件”函数，这个函数将在ode相关命令运行时被调用。

step 3 进行微分方程的求解。在MATLAB的命令窗口中输入下面的代码：

```
%设置微分方程的时间跨度和初始数值
>>tstart = 0;
>>tfinal = 30;
>>y0 = [ 0; 20] ;
%设置进行数据插值
refine = 4;
%设置微分方程的属性
options = odeset('Events',@events,'OutputFcn',@odeplot,'OutputSel',1,...
                 'Refine',refine);
%创建新的图形窗口，并设置对应的属性
figure;
set(gca,'xlim',[ 0 30] ,'ylim',[ 0 22] );
box on
hold on;
tout = tstart;
yout = y0.';
teout = [] ;
yeout = [] ;
ieout = [] ;
for i = 1:10
  % 执行第一次微分方程的求解
  [ t,y,te,ye,ie] = ode23(@bounce,[ tstart tfinal] ,y0,options);
  if ~ishold
    hold on
  end
  % 累积输出结果
  nt = length(t);
  tout = [ tout; t(2:nt)] ;
  yout = [ yout; y(2:nt,:)] ;
  teout = [ teout; te] ;
  yeout = [ yeout; ye] ;
  ieout = [ ieout; ie] ;
```

```
    ud = get(gcf,'UserData');
    if ud.stop
      break;
    end
      % 设置新的积分条件
    y0(1) = 0;
    y0(2) = -.9*y(nt,2);
    % 设置时间步参数的数值
    % 延用了 'refine' 属性值4
    options = odeset(options,'InitialStep',t(nt)-t(nt-refine),...
                            'MaxStep',t(nt)-t(1));
    tstart = t(nt);
end
% 绘制小球撞地的时间轨迹
plot(teout,yeout(:,1),'ro')
% 设置图形的属性
xlabel('time');
ylabel('height');
title('Ball trajectory and the events');
hold off
% 调用函数来绘制图形
odeplot([],[],'done');
grid
set(gca,'Ytick',[0:2:22])
```

step 4 查看程序代码的结果。在输入上面的程序代码后，按"Enter"键，得到的图形如图4.69所示。

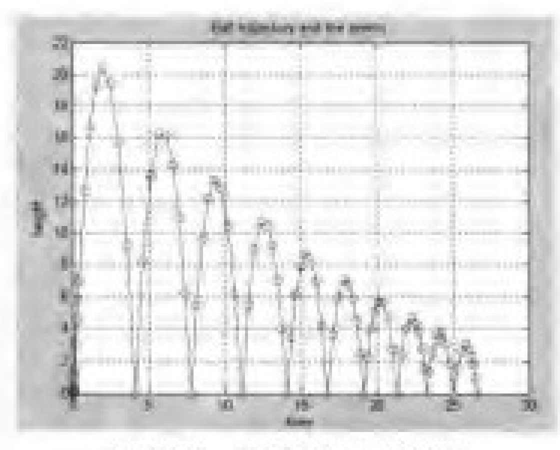


图4.69　小球运动轨迹图形

4.9.5 加权常微分方程

加权常微分方程的通用形式如下：

$$M(t,y)y'=f(t,y) \quad y(t_0)=y_0$$

在上面的方程中，被称为加权函数，是一个矩阵。矩阵中的元素包含了y和t两个变量，并且一般无法转换为一维函数，否则就可以直接转换为右侧的函数表达式。在MATLAB中，求解加权常微分方程的步骤大致如下：

- 编写加权函数 $M(t,y)$ 的M文件；
- 通过odeset命令设置微分方程的"mass"属性；

✦ 选择合适的 ode 类函数来求解微分方程。

在本小节中，将以一个比较简单的例子来说明如何求解加权常微分方程。

例 4.32 求解下面的加权常微分方程，其中

$$M(t,y)=\begin{pmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 1 & m_1+m_2 & 0 & 0 & 0 & -m_2L\sin(y(5))\\ 1 & 0 & 0 & 0 & 0 & 0\\ 1 & 0 & 0 & m_1+m_2 & 0 & m_2L\cos(y(5))\\ 1 & 0 & 0 & 0 & 0 & 0\\ 1 & -L\sin(y(5)) & 0 & L\cos(y(5)) & 0 & L^2\end{pmatrix}$$

同时，微分方程函数如下：

$$f(t,y)=\begin{pmatrix} y(2)\\ m_2L\cdot y^2(6)\cos(y(5))\\ y(4)\\ m_2L\cdot y^2(6)\sin(y(5))-(m_1+m_2)g\\ y(6)\\ -gL\cos(y(5))\end{pmatrix}$$

根据上面的微分方程的参数矩阵，其对应的微分方程组如下：

$$\begin{cases} y(1)=X\\ y(2)=X'\\ y(3)=Y\\ y(4)=Y'\\ y(5)=\theta\\ y(6)=\theta'\end{cases}$$

在上面的方程组中，(X,Y)表示的是物体的坐标数值，θ表示的是物体运动的角度，同时对应的初始数值为 y_0=[1;4;2;20;pi/2;2]。

step 1 选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function M = mass(t,y,m1,m2,L,g)
% 创建新的矩阵，开辟矩阵存储空间
M = zeros(6,6);
M(1,1) = 1;
M(2,2) = m1 + m2;
M(2,6) = -m2*L*sin(y(5));
M(3,3) = 1;
M(4,4) = m1 + m2;
M(4,6) = m2*L*cos(y(5));
M(5,5) = 1;
M(6,2) = -L*sin(y(5));
M(6,4) = L*cos(y(5));
M(6,6) = L^2;
```

在输入上面的程序代码后，将其保存为 "mass.m" 文件，该文件将是在后面的步骤中调用的

加权函数。

step 2 选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function dydt = massode(t,y,m1,m2,L,g)
dydt = [
  y(2)
  m2*L*y(6)^2*cos(y(5))
  y(4)
  m2*L*y(6)^2*sin(y(5))-(m1+m2)*g
  y(6)
  -g*L*cos(y(5))
  ];
```

在输入上面的程序代码后，将其保存为 "massode.m" 文件，该函数代码将是在后面的步骤中需要求解的微分方程组。



在上面的两个函数中，都使用到外部参数 m_1、m_2、L 和 g，这些参数都将在后面步骤中通过程序代码进行赋值。

step 3 在 MATLAB 的命令窗口中输入下面的程序代码：

```
% 定义微分方程的参数
m1 = 0.1;
m2 = 0.1;
L = 1;
g = 9.81;
% 设置微分方程的属性
tspan = linspace(0,4,25);
y0 = [ 0; 4; 2; 20; -pi/2; 2];
options = odeset('Mass',@mass);
% 进行微分方程的求解
[ t y] = ode45(@odemass,tspan,y0,options,m1,m2,L,g);
% 定义绘制的变量数值
theta = y(1,5);
X = y(1,1);
Y = y(1,3);
xvals = [ X X+L*cos(theta)];
yvals = [ Y Y+L*sin(theta)];
% 绘制变量的图形
figure;
plot(xvals,yvals,xvals(1),yvals(1),'ro',xvals(2),yvals(2),'go')
title('A thrown baton problem with mass matrix M(t,y), solved by ODE45');
axis([ 0 22 0 24])
hold on
for j = 2:length(t)
  theta = y(j,5);
  X = y(j,1);
  Y = y(j,3);
  xvals = [ X X+L*cos(theta)];
  yvals = [ Y Y+L*sin(theta)];
  plot(xvals,yvals,xvals(1),yvals(1),'ro',xvals(2),yvals(2),'go')
```

```
end
set(gca,'YLim',[0 24]);
set(gca,'Ytick',[0:2:24]);
grid on
hold off
```

step 4 查看程序代码的结果。在输入上面的程序代码后，按"Enter"键，得到的结果如图4.70所示。

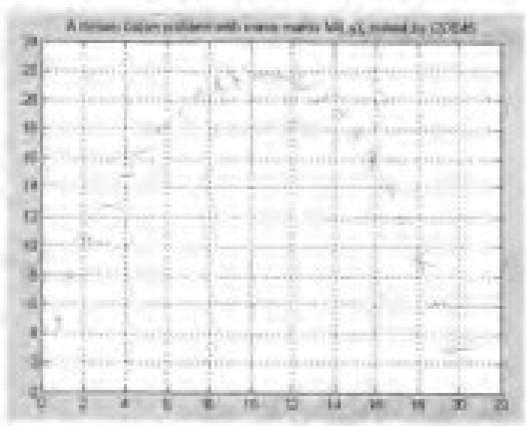


图4.70 微分方程的数值解



在MATLAB中，除了可以解决加权常微分方程（又被称为线性隐式常微分方程）之外，还可以解决完全隐式常微分方程，对应的命令是ode15i，同时设置该隐式常微分方程初始条件的命令是decic，关于这两个命令的详细内容请查看帮助文件。

4.9.6 延迟微分方程命令

在数学理论中，延迟微分方程具有下面的形式：

$$y(t) = f(t, y(t), y(t-\tau_1), \cdots, y(t-\tau_k))$$

在上面的函数表达式中，时间跨度区间为$[t_0, t_f]$，$\tau_1, \cdots, \tau_k$都是常数，表示的是正的时间延迟，同时满足$t_0<t_f$。

在MATLAB中，求解延迟微分方程的命令为dde23，其完整的调用格式如下：

```
sol = dde23(ddefun,lags,history,tspan,options)
```

上面的命令中各参数的具体含义如下。

- ddefun：代表延迟微分方程的M文件函数。其具体的格式为dydt = ddefun(t,y,Z)，其中t代表的是当前时间数值，y是列向量，Z(:,j)代表$y(t-\tau_j)$。
- lags：代表时间延迟向量，其中τ_k保存在变量lags(k)中。
- history：代表y在时间t_0之前的数值，可以使用3种方式来定义history：使用函数$y(t)$来定义y在时间t_0之前的数值；使用一个常数向量来定义y在时间之前的数值，此时y在时间之前的数值被认为是常数；使用前一个时刻的方程解sol来定义y在时间t_0之前的数值。
- tspan：代表时间跨度区间为$[t_0, t_f]$，函数将返回时间段的延迟微分方程的数值解。
- options：关于解法器的参数，可以使用ddeset函数来定义。



说明：上面命令的返回数值sol是一个结构体变量，具有7个属性数值，可以使用相应的程序代码命令来查看具体的属性参数。

4.9.7 延迟微分方程实例

例4.33 求解下面的延迟微分方程组

$$\begin{cases} y_1'(t) = y_1(t-1) \\ y_2'(t) = y_1(t-1) + y_2(t-0.2) \\ y_3'(t) = y_2(t) \end{cases}$$

同时，该方程各变量的历史数据满足$y_1(t)=y_2(t)=y_3(t)=1$，其中$t \leqslant 0$，求解的时间区间为[0,5]，延迟的数据量为[1,0.2]。

step 1 编写微分方程的代码。选择命令窗口编辑栏中的"File"⇨"New"⇨"M-File"命令，打开M文件编辑器，在其中输入下面的程序代码：

```
function dydt = ddex1de(t,y,Z)
% Differential equations function for DDEX1.
ylag1 = Z(:,1);
ylag2 = Z(:,2);
dydt = [ ylag1(1)
         ylag1(1) + ylag2(2)
         y(2)                ];
```

在输入上面的程序代码后，将其保存为"ddex1de.m"文件，该文件将是在后面步骤中调用的微分方程组。

step 2 编写历史数据的代码选择命令窗口编辑栏中的"File"⇨"New"⇨"M-File"命令，打开M文件编辑器，在其中输入下面的程序代码：

```
function s = ddex1hist(t)
% Constant history function for DDEX1.
s = ones(3,1);
```

在输入上面的程序代码后，将其保存为"ddex1hist.m"文件，该文件将是在后面步骤中调用的历史函数。

step 3 求解微分方程。在MATLAB的命令窗口中输入下面的代码：

```
sol = dde23(@ddex1de,[ 1, 0.2],@ddex1hist,[ 0, 5] );
figure;
plot(sol.x,sol.y,'LineWidth',1.5)
title('An example of Wille'' and Baker.');
xlabel('time t');
ylabel('solution y');
grid
legend('Y1','Y2','Y3')
```

step 4 查看程序代码的结果。在输入上面的代码后，按"Enter"键，得到的结果如图4.71所示。

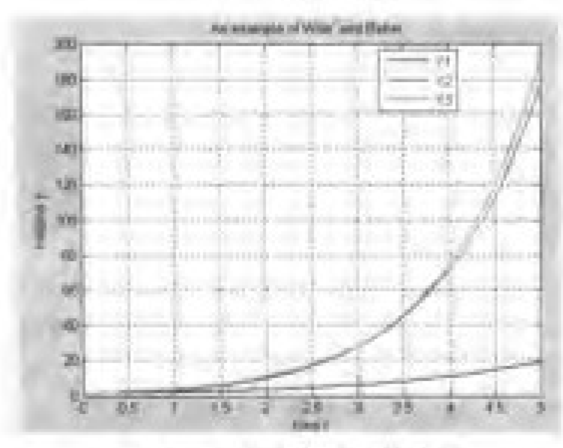


图 4.71 微分方程的数值解



在上面的程序代码中，通过 sol.x 和 sol.y 来引用命令 dde23 求解的数值，然后使用绘图命令绘制对应的结果图形。

4.9.8 常微分方程的边界问题

一般情况下，微分方程的边界问题可能有解、也可能无解，可能有唯一解，也可能有无数解。在本小节中，讨论的问题主要局限在有唯一解的范围内。在有唯一解的情况下，求解边界问题有如下三种基本的方法。

- **迭加法：** 假设微分方程和边界条件都是线性的，问题可以转换为使用 ode 类命令求解的初值问题。
- **试射法：** 问题转换为对漏缺的初值的搜索问题，一旦漏缺的初值被确定，问题就会转换为初值问题。
- **松弛法：** 首先猜测满足边界条件的区间网点上的解值，然后利用微分方程进行迭代改善。

在 MATLAB 中，提供 bvp4c 命令来求解边界问题。该命令使用的是有限元方法，属于上面介绍的松弛法，所得到的精度比较均匀。将常微分方程组整理成下面的形式：

$$\frac{dy}{dx}=f(x,y)$$

同时，满足的边界条件为 $g(y(a),y(b))=0$。其中 a 和 b 是微分方程求解区间的上限和下限，也就是需要求解 y 在 $[a,b]$ 中进行数值求解。

更一般地，MATLAB 还可以求解带有未知参数的微分方程，其具体形式如下：

$$\frac{dy}{dx}=f(x,y,p)$$

其中参数满足的边界条件为 $g(y(a),y(b),p)=0$，上面式子中的变量 p 是未知参数，需要通过边界条件来充分决定。

在 MATLAB 中求解微分方程的相关命令如下：

- solinit = bvpinit(x,yinit,parameters) 生成 bvp4c 命令所必须的猜测数据网格；
- sol = bvp4c(odefun,bcfun,solinit,options) 给出微分方程边界问题的数值解；
- sxint = bvpval(sol,xint) 计算微分方程积分区间内任何一点的解值。

下面详细介绍上面各命令的参数含义。

- 在 bvpinit 命令中，变量 x 的含义是对指定边界区间[*a*,*b*]上的初始网格；yinit 则是对数值解的初始猜测数值，可以是常数向量或者函数向量。当 yinit 是常数向量时，表示解分量中所在所有初始网点上的猜测值都会取 yinit；当 yinit 是函数向量时，表示在解向量所有分量所在的网点上取值为函数值。
- 在命令 bvp4c 中，odefun 表示计算导数的 M 函数文件，其格式为 **dydx=odefun(*x*,*y*)**，或者包含未知参数的格式 **dydx=odefun(*x*,*y*,*p*)**；bcfun 是描述边界条件的函数，取具体格式为 **res=bcfun(*ya*,*yb*,*p*)**，或者包含未知参数的格式 **res=bcfun(*ya*,*yb*,*p*)**；solinit 表示的是对方程解的猜测解，它是一个结构体，包含 *x*、y 和 *p* 三个属性，可以由 bvpinit 命令获得。
- 在命令 bvpval 中，sol 是 bvp4c 的输出变量，xint 则是需要计算区间的参数数值。



命令 bvp4c 的输出结果 sol，同样都是结构体变量，可以使用结构体的相关命令来引用求解的结果，然后进行后续的处理。

4.9.9 边界问题实例

例 4.34 求解下面的微分方程：

$$y''+\frac{2}{x}y'+y^5=0$$

同时该方程的边界条件为 $y'(0)=0$，$y(1)=\frac{\sqrt{3}}{2}$。

根据相关的数学知识，该微分方程有解析解，表达式为 $f(x)=\frac{1}{\sqrt{1+\frac{x^2}{3}}}$，因此在本实例中可以将数值求解得到的数值和该表达式中的数值进行对比。

step 1 编写微分方程的代码。选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function dydx = emdenode(x,y)
%EMDENODE  Evaluate the function f(x,y)
dydx = [   y(2)
          -y(1)^5 ];
```

在输入上面的程序代码后，将其保存为 "emdenode.m" 文件，该文件将是在后面步骤中调用的微分方程组。

step 2 编写边界条件数据的代码选择命令窗口编辑栏中的 "File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function res = emdenbc(ya,yb)
%EMDENBC  Evaluate the residual in the boundary conditions
res = [  ya(2)
         yb(1) - sqrt(3)/2 ];
```

在输入上面的程序代码后，将其保存为 "emdenbc.m" 文件，该文件将是在后面步骤中调用的边界条件函数。

step 3 求解微分方程。在 MATLAB 的命令窗口中输入下面的代码：

```
% 设置微分求解方程
S = [0   0
     0 -2];
options = bvpset('SingularTerm',S);
% This constant guess satisfies the boundary conditions.
guess = [ sqrt(3)/2; 0];
solinit = bvpinit(linspace(0,1,5),guess);
sol = bvp4c(@emdenode,@emdenbc,solinit,options);
% The analytical solution for this problem.
x = linspace(0,1);
truy = 1 ./ sqrt(1 + (x.^2)/3);
% 绘制微分计算的结果
figure;
plot(x,truy,'g','LineWidth',1.5);
hold on
plot(sol.x,sol.y(1,:),'ro');
% 设置图形的其他属性
title('Emden problem -- BVP with singular term.')
legend('Analytical','Computed');
xlabel('x');
ylabel('solution y');
grid
```

step 4 查看代码的结果。输入上面代码后，按"Enter"键，得到的结果如图 4.72 所示。

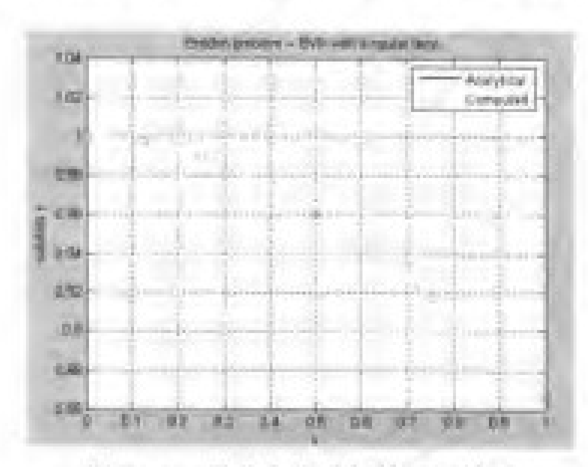


图 4.72　微分方程求解的图形结果



在上面的结果中，使用微分方程求解得到的是数值的唯一解，但是这并不是普遍情况，并不是所有的边界问题都能得到唯一解。

例 4.35　求解下面的微分方程：

$$y''+|y|=0$$

同时该方程满足边界条件为 $y(0)=0$ 和 $y(4)=-2$。

step 1 编写微分方程的代码。选择命令窗口编辑栏中的"File"⇨"New"⇨"M-File"命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function dydx = twoode(x,y)
%TWOODE  Evaluate the differential equations for TWOBVP.
dydx = [  y(2); -abs(y(1)) ];
```

在输入上面的程序代码后，将其保存为"twoode.m"文件，该文件将是在后面步骤中调用的微分方程组。

step 2 编写边界条件函数的代码。选择命令窗口编辑栏中的"File" ⇨ "New" ⇨ "M-File" 命令，打开 M 文件编辑器，在其中输入下面的程序代码：

```
function res = twobc(ya,yb)
%TWOBC  Evaluate the residual in the boundary conditions for TWOBVP.
res = [  ya(1); yb(1) + 2 ];
```

在输入上面的程序代码后，将其保存为"twobc.m"文件，该文件将是在后面步骤中调用的边界条件函数。

step 3 求解微分方程。在 MATLAB 的命令窗口中输入下面的代码：

```
% One solution is obtained using an initial guess of y1(x)=1, y2(x)=0
solinit = bvpinit(linspace(0,4,5),[ 1  0] );
sol = bvp4c(@twoode,@twobc,solinit);
x = linspace(0,4);
y1 = deval(sol,x);
figure;
plot(x,y1(1,:),'r','LineWidth',1.5);
xlabel('x');
ylabel('y');
set(gca,'Ytick',[ -2:0.4:2.4] )
grid on
title('The first solution')
% The other solution is obtained using an initial guess of y1(x)=-1, y2
(x)=0
solinit = bvpinit(linspace(0,4,5),[ -1  0] );
sol = bvp4c(@twoode,@twobc,solinit);
y2 = deval(sol,x);
% Plot both solutions
figure;
plot(x,y1(1,:),'r','LineWidth',1.5);
hold on;
plot(x,y2(1,:),'g','LineWidth',1.5);
set(gca,'Ytick',[ -2:0.4:2.4] )
xlabel('x');
ylabel('solution y');
title('A BVP with two solutions');
grid on;
```

step 4 查看程序代码的结果。在输入上面的程序代码后，按"Enter"键，得到的第一个数值结果如图 4.73 所示。

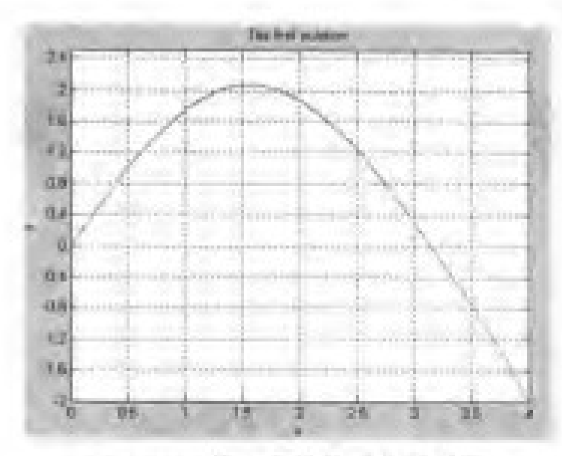


图 4.73　第一个数值求解的结果

由于该微分方程有两个数值解，两个数值解的结果如图 4.74 所示。

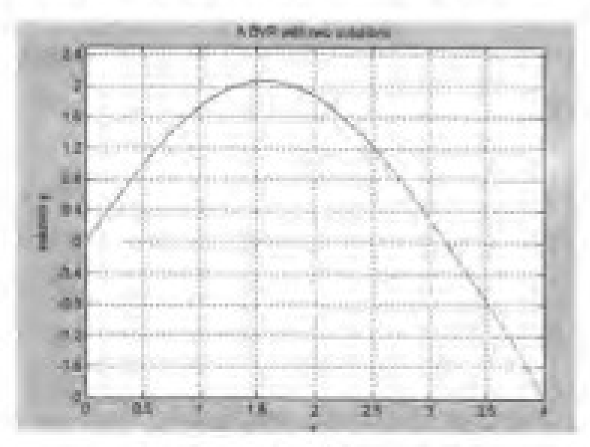


图 4.74　两个数值解的图形



从上面的结果中可以看出，在对应的微分方程和边界条件下，解出了两个满足条件的数值解。

4.10 小结

在本章中依次向读者介绍了如何使用 MATLAB 来进行常见的数据分析：数据插值、曲线拟合、傅里叶分析、优化和常微分方程等，这些应用相对于前面章节的内容而言，更加复杂，涉及的数学原理也比较深入，因此笔者建议读者在阅读本章内容的时候，能够结合数学原理一起理解。在后面的章节中，将介绍如何使用 MATLAB 进行符号计算。

✦　✦　✦

第5章 符号计算

本章包括

- ◆ 符号对象
- ◆ 符号表达式
- ◆ 符号函数
- ◆ 符号微积分
- ◆ 符号代数方程组
- ◆ 符号微分方程
- ◆ 符号积分变换
- ◆ 利用Maple

在前面的章节中，已经介绍MATLAB在数值计算和分析中的应用，MATLAB除了能够处理数值运算之外，还可以进行各种符号计算。在MATLAB中，进行符号计算可以用推理解析的方式进行，避免数值计算带来的截断误差，同时符号计算可以得到正确的封闭解后者精确的数值解。而且，在MATLAB7.0中调用符号计算的命令十分简单，和读者在教科书中使用的公式符号大体相同，可以十分容易地被接受。

在MATLAB中，符号数学工具箱（Symbolic Math Toolbox）中的工具都是建立在数学计算软件Maple的基础上的。如果在MATLAB7.0中进行符号运算，MATLAB会调用MAPLE进行运算然后将结果返回到MATLAB7.0的命令窗口中。正是因为这个原因，当MATLAB7.0进行软件升级或者符号计算内核MAPLE的升级时，符号计算工具箱也随之升级，这些升级内容对于普通用户来说，差别还是比较细微的，对于有比较大变化的地方，在本章的相关位置也会分别给出说明。

在MATLAB中，符号运算实质上属于数值计算的补充部分，并不能算是MATLAB的核心内容。但是，关于符号计算的命令、符号计算结果的图形显示、计算程序的编写或者帮助系统等，都是十分完整和便捷的。

 符号对象和符号表达式

在科学工程中，数值运算是十分重要的内容，但是自然科学理论中，各种公式、表达式以及相应的推导等也是十分重要的，这些就是符号运算解决的重点内容。在MATLAB中，数值和数值变量用于数值的存储和各种计算，而符号对象、变量、函数以及相应的操作都是用来形成符号表达式的，然后按照相关数学内容的规则进行运算，得出相应的解析解。

 sym命令

和在MATLAB中使用数值计算一样，使用数值表达式的变量必须首先进行变量赋值，否则MATLAB会返回变量错误信息。符号数学工具箱也沿用该规则，在进行符号运算之前，首先需要定义符号对象，然后利用这些符号对象去构建表达式，最后才能进行符号运算。

从上面的介绍中可以看出，创建符号对象是进行符号运算的基础，MATLAB提供多种创建符号对象的命令。数值、字符串、符号对象是MATLAB中常见的三种变量，MATLAB提供将数值或者字符串变量转化为符号对象的方法，同时提供将符号对象转换成为数值或者字符串变量的方法。

本小节首先介绍如何创建符号对象，关于其他的转换方法将在后面小节中详细介绍。定义符号

对象的常见命令是sym，常见的调用格式如下：

```
Sym1=sym(argn,flagn)
Sym2=sym('argv',flagv)
```

在上面的调用格式中，使用sym命令创建了符号对象Sym1和Sym2。下面详细介绍这两种调用格式的含义。

Sym1=sym(argn,flagn) 将数值或者数值表达式转换为符号对象Sym1，参数flagn的作用是定义转换的符号对象应该符合的格式，其具体的选项和含义如下：

- 'd' 用最接近的十进制浮点精确表示；
- 'e' 当表示数值计算时，带估计误差的有理表示；
- 'f' 用十六进制浮点表示；
- 'r' 这是MATLAB的默认设置，用最接近有理表示的形式。

表达式Sym2=sym('argv',flagv)将指定的字符串变量argv转换为符号对象Sym2，参数flagv的作用也是定义转换的符号对象应该符合的格式，其具体的选项和含义如下：

- 'positive' 限定A表示为正的实型符号变量；
- 'real' 限定A为实型符号变量；
- 'unreal' 限定A为非实型符号变量。

使用sym命令创建符号对象

例5.1 将数值或者数值表达式转化为符号对象，使用不同的转换格式。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> sym1=sym([ 3/4,log(10),exp(2),log(10)+exp(2)],'d');
>> sym2=sym([ 3/4,log(10),exp(2),log(10)+exp(2)],'e');
>> sym3=sym([ 3/4,log(10),exp(2),log(10)+exp(2)],'f');
>> sym4=sym([ 3/4,log(10),exp(2),log(10)+exp(2)],'r');
>> my_sym=[ sym1;sym2;sym3;sym4]
```

step 2 按"Enter"键，就会得到上面程序代码的结果。

```
my_sym =

[ .75000000000000000000000000000000, 2.3025850929940459010936137929093,
7.3890560989306504069418224389665, 9.6916411919246954198570165317506]
[                    3/4,          5184960683398422*2^(-51),
8319337573440942*2^(-50),          5455908957570076*2^(-49)]
[          '1.8000000000000'*2^(-1),          '1.26bb1bbb55516'*2^(1),
'1.d8e64b8d4ddae'*2^(2),           '1.3621ecb57c41c'*2^(3)]
[                    3/4,          5184960683398422*2^(-51),
8319337573440942*2^(-50),          5455908957570076*2^(-49)]
```

在上面的程序中，分别将数值3/4，数值表达式log(10)、exp(2)等转换为符号变量，差别在于不同的转换格式，根据上面的结果可以看出转换格式对结果的影响。

step 3 查看程序结果的信息。在上面的结果中，sym1～sym4都是符号变量，而my_sym则是符号

矩阵，可以在MATLAB中输入whos来查看这些变量的类别和大小：

```
>> whos
  Name          Size                            Bytes  Class
  my_sym        4x4                              1774  sym object
  sym1          1x4                               568  sym object
  sym2          1x4                               454  sym object
  sym3          1x4                               490  sym object
  sym4          1x4                               454  sym object
Grand total is 782 elements using 3740 bytes
```



在本章节前面的部分，已经介绍MATLAB将数值或者表达式转换为符号变量时，默认格式是r。因此，如果用户不使用格式转换参数，得到的结果就是变量sym4。上面的结果并不存在实质上的优劣，用户应该根据需要选择转换的格式。

例5.2 在MATLAB中，使用符号变量验证积分等式 $\int_a^b \frac{1}{x}\mathrm{d}x = \ln\frac{b}{a}$。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> Var=sym('var','positive');                 % 定义正的积分变量
>>Upper=sym('upper','real');                  % 定义积分上限
>>Lower=sym('lower','real');                  % 定义积分下限
>>Integral= int(1/(Var),Lower,Upper)          % 计算积分数值
```

step 2 在MATLAB的命令窗口中输入"Integral"，然后按"Enter"键，就会得到上面程序代码的结果：

```
Integral =
 log(upper)-log(lower)
```

在MATLAB中，对数Ln用自然对数Log表示。因此，上面程序代码得到的结果就是 $\log(upper)-\log(lower)=\ln(upper)-\ln(lower)=\ln\frac{upper}{lower}$。整个程序代码相当于下面的等式：$\int_{lower}^{upper}\frac{1}{x}\mathrm{d}x=\ln\frac{upper}{lower}$，因此也就证明了积分等式 $\int_a^b\frac{1}{x}\mathrm{d}x=\ln\frac{b}{a}$。

由于积分结果是对数函数，为了让最后的结果在数学上有意义，在上面的程序代码中将积分变量定义为正的实型符号变量，因此格式参数为positive。而积分上下限则是实数型的，因此格式参数为real。



本实例中，使用积分符号int来进行符号运算，在MATLAB中还提供其他运算符号来进行符号运算。这些符号和在经典教科书中的符号大致类似，这些内容将在后面详细介绍。

例5.3 在MATLAB中，使用不同的格式输入符号变量，并比较不同格式间的差异。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> s1=sym('[ 3/7,exp(1),exp(2)+log(4)] ');
>> s2=sym('[ 3/7 exp(1) exp(2)+log(4)] ');
>> s3=sym([ 3/7,exp(1),exp(2)+log(4)] );
>> sd=[ s1;s2;s3] ;
```

step 2 在MATLAB的命令窗口中输入"sd"，然后按"Enter"键，就会得到上面程序代码的结果：

```
>> sd
 sd =
[              3/7,                exp(1),        exp(2)+log(4)]
[              3/7,                exp(1),        exp(2)+log(4)]
[              3/7, 6121026514868074*2^(-51), 4940083132741141*2^(-49)]
```

在上面的程序代码中，显示了在MATLAB中定义符号变量的常用方法，建议读者使用本例中变量S1的创建方法，这是因为这种创建方法在任何版本的符号工具箱都适用，而且产生的符号数值表示是绝对准确的。



尽管从上面的演示结果来看，使用第一种和第二种方法创建的结果相同，但是还是建议使用第一种方法，因为使用符号计算比较占用计算机的资源，第一种创建方法比较节省资源。

5.1.3 使用syms命令创建符号对象

除了上面小节中介绍的sym命令之外，MATLAB还提供syms命令来创建符号对象。这个命令是sym命令的快捷方式，使用起来比sym命令更加简洁，可以同时将多个变量创建为符号对象。

在MATLAB中，syms命令的常见调用格式如下：

```
syms arg1 arg2 ... flagv
```

在上面的调用格式中，arg1、arg2...表示变量，可能是数值变量也可能是字符串变量，而flagv表示转换格式，和sym命令相同。

例5.4 使用syms命令创建符号变量。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> clear all
>> syms alpha beta thro real;
>> whos
```

step 2 按"Enter"键，就可以得到相应的结果如下：

```
>> whos
  Name        Size                    Bytes  Class
  alpha       1x1                       134  sym object
  beta        1x1                       132  sym object
  thro        1x1                       132  sym object
 Grand total is 16 elements using 398 bytes
```

从上面的结果可以看出，使用syms命令可以同时将三个变量设置为符号变量。为了方便读者理解这种命令，上面的代码syms alpha beta thro real相当于下面的几行代码：

```
>> alpha=sym('alpha','real');
>> beta=sym('beta','real');
```

```
>> thro=sym('thro','real');
```

这两种代码得到的结果是完全相同的。如果灵活使用syms命令，就可以省略前面小节的烦琐命令，这符合 MATLAB 符号运算简洁的特点。



在上面的程序语句中，第一行代码clear all的作用是删除内存中储存的所有变量，这样在程序的最后可以使用whos命令查看由syms创建的符号变量。

使用sym和syms都可以创建符号对象或者符号变量，下面介绍MATLAB中关于符号变量的命令规则。符号变量也是MATLAB中的常见变量类型，其命令规则和数值变量规则相同，主要规则如下：

- 变量名可以由英文字母、数字或者下划线组成；
- 变量名必须用英文字母开始；
- 组成变量名称的字符长度不能超过31；
- MATLAB会区别变量名称的大小写。

5.1.4 符号计算的运算符和函数

前面小节已经介绍了如何在 MATLAB 中创建符号对象，但是，仅仅创建符号对象还不能利用 MATLAB 中的符号资源，如果希望使用 MATLAB 解决更丰富的符号问题，则需要创建符号表达式。

和数值表达式一样，构成符号表达式的基础元素是运算符和函数。无论在形式、名称上，还是在使用方式上，符号运算的运算符与函数和数值计算几乎完全相同，这些相同的地方会给用户在 MATLAB 编程上带来极大的方便。

下面简单介绍符号计算中的运算符和函数的使用方法：

- 基础运算符
  对于加、减、乘、左除、右除、求幂等计算，符号运算和数值运算的符号和使用方法完全相同。同时，对MATLAB 中特有的点乘、点除等特殊计算方式，符号运算同样支持。
- 关系运算符
  MATLAB的符号运算提供两种关系运算符："==" 和 "~="，分别表示两个符号对象相等和不等。关系运算符返回的是逻辑值 1 和 0，1 表示关系运算成立，0 则表示关系运算不成立。
- 三角、双曲函数
  在MATLAB中，除了函数atan2函数只能用在数值运算中之外，所有的三角函数、双曲函数以及对应的反函数，都可以用在符号运算中。
- 指数、对数函数
  在 MATLAB 中，指数函数可以通用于数值运算和符号运算中。但是对于对数函数，在符号运算中只能使用 log 函数，而不能使用 log2 和 log10 函数。
- 复数函数
  在 MATLAB 中，关于复数的共轭、实部、虚部和求模等，在符号运算和数值运算中完全相同。在符号运算中，没有提供求相角的函数。
- 矩阵代数命令
  在 MATLAB 中，关于矩阵代数命令在数值运算和符号运算中几乎完全相同，只是关于求解奇异解的 svd 命令有所不同，将在后面章节中详细介绍。



在创建符号表达式之前，有必要了解构建表达式的符号和函数，这样就不会在后面的步骤中造成不必要的影响。例如不应该使用 ln 函数而应该使用 log 函数，否则会返回错误信息。

5.1.5 识别对象的命令

在MATLAB中，数值对象、字符串对象和符号对象都是常见的对象类型，都分别有着不同的运算规则，但有时这些不同的对象类型在外观上没有太大的区别。为了避免用户在使用过程中造成混淆，MATLAB提供了识别不同对象类型的命令。

常见的识别对象命令有class、isa和whos等，用户可以根据需要来选择不同的命令，识别不同的对象。下面将使用一个简单的案例来说明如何使用这些命令。

例 5.5 使用MATLAB命令来识别不同的对象类型。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> clear all
>> my_number=[pi,sqrt(5),exp(4)];
>> my_char='[pi,sqrt(5),exp(4)]';
>> my_sym=sym(my_char);
```

在上面的程序代码中，分别创建了数值、字符串和符号对象。这样，在后面的步骤中可以使用命令来识别不同的对象。

step 2 使用class命令来判断不同的对象类型，在命令窗口中输入下列内容：

```
>> Var_class=char(class(my_number),class(my_char),class(my_sym))
Var_class =
double
char
sym
```

在上面的程序代码中，使用class命令来判断上面步骤中创建的对象类型，得到的结果是doule（双精度）、char（字符串）、sym（符号对象）。



通过class命令判断的对象类型结果是字符串变量。也就是说，class(my_number)返回的结果是字符串"double"。为了创建字符串数组需要使用char命令。

step 3 使用isa命令来判断不同的对象类型，在命令窗口中输入下列内容：

```
>> Isa_var=[isa(my_number,'double'),isa(my_char,'char'),isa(my_sym,
'sym')];
>> Isa_var
Isa_var =

     1     1     1
```

在上面的程序中，通过使用isa来判断变量的类型，返回逻辑值1说明上面程序的判断是正确的。也就是说，my_number是双精度（double）类型，my_char是字符串（char）类型，

my_sym是符号（sym）类型。

step 4 使用whos命令来判断不同的对象类型，在命令窗口中输入下列内容：

```
>> whos my_char my_number my_sym
  Name              Size                       Bytes  Class
  my_char           1x19                          38  char array
  my_number         1x3                           24  double array
  my_sym            1x3                          274  sym object
Grand total is 40 elements using 336 bytes
```

在上面的程序中，之所以输入的命令是“whos my_char my_number my_sym”，而不是直接输入命令“whos”，是因为在前面的步骤中创建了其他变量；为了简化查询结果，就直接查询上面三个变量的类型。

确定符号表达式中的变量

为了简化符号对象的操作和计算，MATLAB为用户提供了findsym命令，可以实现对符号表达式中符号变量或者指定数目的变量的自定认定。

其常见的调用格式如下：

- r = findsym(S) 确定符号表达式或者矩阵 S 中自由符号变量；
- r = findsym(S,n) 确定符号表达式或者矩阵 S 中靠x最近的n个独立符号变量。

例5.6 使用MATLAB的命令来确定符号表达式中的变量。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> syms a x y z t
```

step 2 确定下面简单符号表达式中的符号变量信息：

```
>> findsym(sin(pi*t))
ans =
t
```

step 3 确定下面简单符号表达式中的符号变量信息：

```
>> findsym(x+i*y-j*z)
ans =
x, y, z
```

step 4 确定下面简单符号表达式中的符号变量信息：

```
>> findsym(a+y,1)
ans =
y
```

从上面的简单实例中可以看到findsym函数的使用方法及其使用范围，在后面的使用步骤中可以灵活使用该命令。

 符号精度计算

前面已经介绍符号计算的一个重要特点就是计算过程中不会出现舍入误差，可以得到任意精度的计算。也就是说，如果希望计算结果精确，那么就应该牺牲计算时间和存储空间，使用符号计算来得到足够高的计算精度。

在 MATLAB 的符号计算工具箱中，提供了如下三种不同类型的计算精度。

- **数值类型**：MATLAB7.0 中的浮点算术计算。
- **有理数类型**：MAPLE 中的精确符号计算。
- **VPA 类型**：MAPLE 的任意精度算术计算。

上面三种不同的运算方式各有利弊，读者需要在使用过程中根据计算精度、时间和存储空间的要求，选用不同的计算精度。

例 5.7 在 MATLAB 中，使用不同的精度计算数学表达式。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> format long
>>var_float=1/3+1/5;
>>var_char=sym(1/3+1/5);
```

step 2 在 MATLAB 的命令窗口中依次输入 var_float、var_char 查看计算结果如下：

```
>> var_float
var_float =
   0.53333333333333
>> var_char
var_char =
 8/15
```

step 3 使用 whos 命令查看不同结果的数据类型，得到的结果如下：

```
>> whos var_float var_char
  Name            Size                  Bytes  Class
  var_char        1x1                     132  sym object
  var_float       1x1                       8  double array
Grand total is 6 elements using 140 bytes
```

从上面的运算结果中可以看出，浮点运算结果 var_float 并不精确，但是计算所占的内存最小，而且运算速度最快。在 MATLAB7.0 中，双精度输出的数字位数由 format 命令控制。在上面的运算过程中，存在着三种计算误差：第一种误差来自于由计算 1/3 的除法舍入误差，第二种误差来自于相加得到的舍入误差，第三种误差来自于二进制转换为十进制的结果。

而另外一面，符号运算结果 var_char 则十分精确，但是所需要的时间和占用的内存资源都是很大的。而且，通过最后的命令可以看出，符号计算得出的结果都是字符串，尽管从形式上是数值，但是在变量类型上还是字符串。

如果需要从精确解中获得任意精度的解，并改变默认的精度，把任意精度符号解变成数值解，需要使用 MATLAB7.0 提供了命令：Vpa 和 digits，其相应命令的调用格式如下：

- digits(d) 设置今后数值计算以d位相对精度进行；
- digits 显示当前采用的数值计算精度；
- R = vpa(A) 在digits指定的精度下，给出A的数值型符号结果R；
- R = vpa(A,d) 在d位相对精度下，给出A的数值型符号结果R。



除了使用vpa(A,d)来指定精度之外，所有vpa(A)的精度都会受前面的digits命令控制，其默认精度是32。而且其变量A既可以是符号对象，也可以是数值对象，但是命令运行后得到的结果都是符号对象。

例 5.8 在MATLAB中，显示不同的显示精度。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> s1=sym('(exp(4)+log(2))');
>> s2=sym(exp(4)+log(2));
>> v1=vpa(abs(s1-s2));
>> v2=vpa(abs(s1-s2),64);
```

step 2 在MATLAB中输入"[s1;s2;v1;v2]"后，按"Enter"键，查看上面运算的结果。

```
>> [s1;s2;v1;v2]
 ans =
                                              (exp(4)+log(2))
                                      7781558297764672*2^(-47)
                                        .1009969127860108e-14
 .1009969127860108544716959987172974323979947273970e-14
```

step 3 最后，在MATLAB中输入"digits"后，按"Enter"键，查看结果如下：

```
>> digits
 Digits = 32
```



尽管在命令行v2=vpa(abs(s1-s2),64)中将精度改为64，但是这个命令并没有改变全局的符号数值，因此digits命令得到的结果还是默认的数值32。

5.3 符号表达式的操作

从前面的小节中，读者可以发现符号运算的结果一般比较繁杂，不直观。为此，MATLAB为用户提供处理符号表达式和函数的操作命令，例如因式分解、展开、简化等，这些命令都可以增加符号表达式的可读性。这些命令的共同点是它们完成的都是符号表达式的恒等变换。MATLAB提供多个函数collect、expand、factor、horner等，灵活使用这些命令，可以增加符号表达式结果的可读性。下面将分别详细介绍这些函数的用法和应用。

5.3.1 Collect 函数

在MATLAB中，collect函数的调用格式如下：

- R = collect(S) 将表达式 S 中相同次幂的项合并，S 可以是表达式或者符号矩阵。
- R = collect(S,v) 将表达式 S 中 v 的相同次幂项合并，v 的默认值是 x。

该函数实现的功能是符号表达式中的同类项合并。

例5.9 在MATLAB中，按照不同的方式合并表达式$(x+e^{-y}x^3-y)*(\sqrt{x}y+e^{-2y}x)$中的参数同类项。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> Exper=sym('(x+exp(-y)*x^3-y)*(sqrt(x)*y+exp(-2*y)*x)');
>> R1=collect(Exper,x);
>> R2=collect(Exper,y);
>> R3=collect(Exper,exp(-y));
```

step 1 在 MATLAB 中输入 "R=[R1;R2;R3]" 后，按 "Enter" 键，查看使用不同方式合并的结果如下：

```
>> R=[ R1;R2;R3]
 R =
  exp(-y)*exp(-2*y)*x^4+exp(-y)*y*x^(7/2)+exp(-2*y)*x^2+y*x^(3/2)
  -y*exp(-2*y)*x-x^(1/2)*y^2-x^(1/2)*y^2+((x+exp(-y)*x^3)*x^(1/2)
  -exp(-2*y)*x)*y+(x+exp(-y)*x^3)*exp(-2*y)*x
  x^3*(x^(1/2)*y+exp(-2*y)*x)*exp(-y)+(x-y)*(x^(1/2)*y+exp(-2*y)*x)
```

step 3 在 MATLAB 中输入 "RM=collect(Exper)" 后，查看 MATLAB 默认情况下的合并类型，得到的结果如下：

```
>> RM=collect(Exper)
 RM =

exp(-y)*exp(-2*y)*x^4+exp(-y)*y*x^(7/2)+exp(-2*y)*x^2+y*x^(3/2)
-y*exp(-2*y)*x-x^(1/2)*y^2
```

step 4 重新输入表达式，将 Exper 表达式中的符号改为 w、t，查看 MATLAB 合并的不同的结果，得到的结果如下：

```
>> Exper1=sym('(w+exp(-t)*w^3-t)*(sqrt(w)*t+exp(-2*t)*w)');
>> R4=collect(Exper1)
 R4 =

exp(-t)*exp(-2*t)*w^4+exp(-t)*t*w^(7/2)+exp(-2*t)*w^2+t*w^(3/2)
-t*exp(-2*t)*w-t^2*w^(1/2)
```

step 1 分析上面的计算结果：

从上面的步骤（2）中的结果可以看出，当使用不同的合并条件时，同样的符号表达式会得出不同的结果，因此在实际应用中应该根据需要选择不同的合并条件；

从上面的步骤（3）中的结果可以看出，MATLAB 的默认合并条件是按照变量 x 进行合并的，因此 RM 和 R1 得到的表达式完全相同；

从步骤（4）中的结果可以看出，当表达式中没有变量 x 的时候，MATLAB 会按照首先出现的符号变量进行同类项合并。



合并同类项的函数collect功能只能按照参数进行合并，结果未必比原来的表达式更加简单。如果希望简化某表达式，还需要使用其他简化命令。

5.3.2 Expand 函数

在 MATLAB 中，expand 函数的功能是将符号表达式进行展开，其调用格式如下：

```
R = expand(S)
```

在上面的参数中，表达式S中如果包含函数，MATLAB会利用恒等式变形将其写成相应的和形式。这个展开函数主要用于多项式、三角函数、指数函数和对数函数。

例 5.10 在 MATLAB 中，使用 expand 对各种类型函数进行展开。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> syms alpha theta a b x y
>> R1=expand(tan(aplha+theta));
>> R2=expand(2^(x+y));
>> R3=expand((a+b)^7);
```

step 2 在 MATLAB 中输入 "R=[R1;R2;R3]" 后，按 "Enter" 键，查看使用不同方式合并的结果如下：

```
>> R=[ R1;R2;R3]
 R =
                         (tan(aplha)+tan(theta))/(1-tan(aplha)*tan(theta))
                                                                   2^x*2^y
 a^7+7*a^6*b+21*a^5*b^2+35*a^4*b^3+35*a^3*b^4+21*a^2*b^5+7*a*b^6+b^7
```

从上面的实例可以看出，MATLAB中的expand函数可以使用多项式、三角函数、指数函数或者对数函数中的恒等式进行展开，得到关于各个符号变量的最简表达形式。

其中，前面两个等式分别来自于下面的恒等式：

$$\tan(\alpha+\beta)=\frac{\tan\alpha+\tan\beta}{1-\tan\alpha\tan\beta}$$

$$2^{x+y}=2^x\times2^y$$



展开函数expand同样有其使用范围，MATLAB并不能解决所有的恒等式变换的问题，除了上面例子中演示的函数类型外，MATLAB将无法处理其他复杂的恒等式。

5.3.3 Factor 函数

在 MATLAB 中，factor 函数的功能是将符号多项式进行因式分解，它的调用格式如下：

```
factor(S)
```

其中，S是多项式或者多项式矩阵，系数是有理数，MATLAB会将表达式S表示成系数为有理数的低阶多项式相乘的形式；如果多项式 S 不能进行在有理数范围内因式分解，该函数会返回 S 本身。

例5.11 在MATLAB中，演示如何在有理数范围内对多项式x^n-y^n进行因式分解，其中n是[1，8]范围内的整数。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> syms x y real
>> n=1:8;
>> p=x.^n-y.^n;
>> f=[p;factor(p)]';
```

step 2 在MATLAB的命令窗口中输入"f"，然后按"Enter"键，就可以得到相应的结果如下：

```
f =

[                    x-y,                                             x-y]
[                x^2-y^2,                                   (x-y)*(x+y)]
[                x^3-y^3,                           (x-y)*(x^2+x*y+y^2)]
[                x^4-y^4,                           (x-y)*(x+y)*(x^2+y^2)]
[                x^5-y^5,             (x-y)*(x^4+x^3*y+x^2*y^2+x*y^3+y^4)]
[                x^6-y^6,       (x-y)*(x+y)*(x^2+x*y+y^2)*(x^2-x*y+y^2)]
[                x^7-y^7, (x-y)*(x^6+x^5*y+x^4*y^2+x^3*y^3+x^2*y^4+x*y^5+y^6)]
[                x^8-y^8,               (x-y)*(x+y)*(x^2+y^2)*(x^4+y^4)]
```

从上面的结果中可以看出，MATLAB中的factor命令是在有理数范围内进行因式分解的，例如，$x^8-y^8=(x-y)(x+y)(x^2+y^2)(x^4+y^4)$分解的结果中因子$(x^4+y^4)$在有理数范围内无法继续分解，但是在实数范围内却可以继续分解。



上面的实例演示了factor函数在多项式中的应用，除了可以在多项式中有着强大的分解能力，factor函数还可以在正整数质数分解中广泛地应用，可以使用该命令对实数进行质数分解。

例5.12 在MATLAB中对正整数系列$f(n)=10^n-1$进行质数分解，其中n是[1，9]范围内的整数。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> syms p real;
>> n=1:9;
>> p=sym(10.^n-1);
>> f=conj([p;factor(p)]');
```

step 2 在MATLAB的命令窗口中输入"f"，然后按"Enter"键，就可以得到相应的结果如下：

```
>> f
 f =
[                 9,                              (3)^2]
[                99,                        (3)^2*(11)]
[               999,                        (3)^3*(37)]
[              9999,                 (3)^2*(11)*(101)]
[             99999,                 (3)^2*(41)*(271)]
[            999999,     (3)^3*(7)*(11)*(13)*(37)]
[           9999999,               (3)^2*(239)*(4649)]
[          99999999,   (3)^2*(11)*(73)*(101)*(137)]
[         999999999,               (3)^4*(37)*(333667)]
```

在上面的程序代码中，使用 factor 命令求解了上面所有数值的质数分解结果，最后一行程序代码中则使用了 conj 命令求得复数的共轭。



实际上，具有整数分解功能的函数 factor 并不是属于符号工具箱（Symbolic Math Toolbox）中的，而是属于特殊数学函数工具箱（MATLAB Function）中的，但是两个函数名称都是 factor。因此，当用户使用该函数时，MATLAB 会调用数学函数工具箱（MATLAB Function）中的 factor 函数。

5.3.4 Horner 函数

在 MATLAB 中，horner 函数的功能是将符号表达式转换成为嵌套形式，调用格式如下：

```
R = horner(P)
```

其中，P 是符号多项式矩阵，函数 horner 将其中每个多项式转换成它们各自的嵌套形式。

例 5.13 在 MATLAB 中演示对多项式的嵌套分解。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> syms t;
>> p=t^8-7*t^7+5*t^6-10*t^5+8*t^4-4*t^3+3*t^2+5*t-8;
>> r=horner(p);
```

step 2 在 MATLAB 的命令窗口中输入"r"，然后按"Enter"键，得到的结果如下：

```
>> r
 r =

-8+(5+(3+(-4+(8+(-10+(5+(-7+t)*t)*t)*t)*t)*t)*t)*t
```



horner 函数完成的结果并不是多项式的因式分解结果，而只是将多项式化解为符号变量的嵌套格式。如果希望完成因式分解，则需要使用 factor 命令。

例 5.14 在 MATLAB 中演示对多变量的多项式的嵌套分解。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> syms x y;
>> p1=x^2+x+y^3-2*y;
>> p2=x^2+x+y^3-2*y+x*y-x^2*y^2;
>> r1=horner(p1); r2=horner(p2);
>> R=[ r1;r2] ;
```

step 2 在 MATLAB 的命令窗口中输入"R"，然后按"Enter"键，得到的结果如下：

```
>> R
 R =

                (-2+y^2)*y+(1+x)*x
 (-2+y^2)*y+(y+1+(1-y^2)*x)*x
```



从上面的结果中可以看出，当输入的多项式中包含着多个变量时，MATLAB会对每个变量进行嵌套分解，将第一个变量分解完成以后，再对下一个变量进行嵌套分解。

5.3.5 Numden 函数

在MATLAB中，numden函数的功能是提取表达式的最小分母公因式和分子多项式，其调用格式如下：

```
[N,D] = numden(A)
```

其中，A 是符号多项式，N(Numerator)是计算得到的最小分母公因式，D(Denominator)则是计算的相应分子多项式。

例 5.15 在 MATLAB 中计算多项式$\frac{x^2-1}{x+2}+\frac{2x+5}{3x-2}$的分母和分子。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> syms x
>> A=(x^2-1)/(x+2)+(2*x+5)/(3*x-2);
>> [n,d]=numden(A)
```

step 2 查看运算的结果。按"Enter"键，就可以得到相应的结果如下：

```
n =

3*x^3+6*x+12

d =

(x+2)*(3*x-2)
```

step 3 分析计算结果。根据基础数学知识，上面两个分式相加的结果如下：

$$\frac{x^2-1}{x+2}+\frac{2x+5}{3x-2}=\frac{3x^3+6x+12}{(x+2)(3x-2)}$$

因此，上面的结果中 n 就是结果中的分母部分，d 就是分子部分。

5.3.6 Simplify 函数

在 MATLAB 中，simplify 函数的功能是根据一定规则对符号表达式进行简化，其调用格式如下：

```
R = simplify(S)
```

其中，S 表示的是符号表达式或者矩阵，R 是经过简化后的符号表达式。



该函数是一个强有力的具有普遍意义的工具，它可以用于指数、对数、三角函数等各种数学表达式中，MATLAB 会利用 MAPLE 化的简化规则对表达式进行简化。

例 5.16 在 MATLAB 中使用 simplify 函数简化符号表达式。

step 1 在命令窗口输入以下内容：

```
>> syms a b positive
>> syms t x real
>> R1=simplify(sqrt(a^2+2*a*b+b^2));
>> R2=simplify(2^a*2^b);
>> R3=simplify(sec(t)^2-tan(t)^2);
>> R4=simplify((x^3-1)/(x-1));
>> R=[R1;R2;R3;R4];
```

step 2 在MATLAB的命令窗口中输入"R"，然后按"Enter"键，就可以得到对应的结果如下：

```
>> R
R =
      a+b
 2^(a+b)
        1
 x^2+x+1
```



从上面的结果中可以看出，在MATLAB中使用simplify函数可以完成各种类型的恒等变换，包括指数、对数和三角变换等。

5.3.7 Simple 函数

在MATLAB中，simple函数的功能是运用包括simplify在内的各种命令将符号表达式转换成为最简洁的形式，其常见的调用格式如下：

- r = simple(S) 该函数公式的功能是使用不同的变换简化规则来对符号表达式进行简化，返回表达式S的最简形式。如果S是符号表达式矩阵，则返回表达式矩阵变成最短的形式，而不一定是使每一项都最短。如果不给定输出参数r，该函数将显示所有使表达式S变短的最简化形式，并返回其中最短的一个表达式。
- [r,how] = simple(S) 不显示简化的中间结果，只是显示寻找到的最短形式以及所有可以使用的简化方法。r是符号表达式的结果，how则是使用的方法。

函数simply的目标是使符号表达式最短，使用最少的字符来表示，使用simply函数通常能够得到理想的结果，为了达到这种要求，simply函数会根据情况调用不同的简化方法，表5.1列出了常见的简化方法。

表 5.1 simply 调用的简化方法

符号表达式	简化结果	调用的方法
cos(x)^2+sin(x)^2	1	simplify
2*cos(x)^2-sin(x)^2	3*cos(x)^2-1	simplify
cos(x)^2-sin(x)^2	cos(2*x)	combine
cos(x)+ (-sin(x)^2)^(1/2)	cos(x)+i*sin(x)	radsimp
cos(x)+i*sin(x)\	exp(i*x)	convert
(x+1)*x*(x-1)	x^3-x	collect
x^3+3*x^2+3*x+1	(x+1)^3	factor
cos(3*acos(x))	4*x^3-3*x	expand

上面的调用方法，有些已经在前面介绍过，有些则没有介绍过，下面简单介绍simply函数常用

调用函数的使用环境：

- 函数 radsimp 用来处理包含根式的表达式；
- 函数 combine 将表达式中以求和形式、乘积形式或者幂形式出现的各项进行合并；
- 函数 convert 将一种形式转换成另一种形式。

例 5.17 在 MATLAB 中使用 simple 命令化简表达式 $\sqrt{\frac{1}{x^2}+\frac{1}{(x-1)^2}-2\times\frac{1}{x(x-1)}}$。

step 1 在命令窗口输入以下内容：

```
>> syms x positive
>> f=sqrt(1/x^2+1/(x-1)^2-2*1/((x-1)*x));
```

step 2 如果希望查看MATLAB使用simple命令调用的所有相关命令，可以在命令窗口中输入"simple(f)"，然后按"Enter"键，得到的结果如下：

```
>> simple(f)
 simplify:
 1/x/abs(x-1)
 radsimp:
 1/(x-1)/x
 combine(trig):
 1/x/((x-1)^2)^(1/2)
 factor:
 (1/x^2/(x-1)^2)^(1/2)
  expand:
 (1/x^2+1/(x-1)^2-2/(x-1)/x)^(1/2)
  combine:
 (1/x^2+1/(x-1)^2-2/(x-1)/x)^(1/2)
  convert(exp):
 (1/x^2+1/(x-1)^2-2/(x-1)/x)^(1/2)
 convert(sincos):
 (1/x^2+1/(x-1)^2-2/(x-1)/x)^(1/2)
  convert(tan):
 (1/x^2+1/(x-1)^2-2/(x-1)/x)^(1/2)
  collect(x):
 (1/x^2+1/(x-1)^2-2/(x-1)/x)^(1/2)
  mwcos2sin:
 (1/x^2+1/(x-1)^2-2/(x-1)/x)^(1/2)
  ans =
 1/(x-1)/x
```



从上面的结果中可以看出，MATLAB 会调用所有相关的命令，分别得出化简的结果，然后比较所有的化简结果，选择其中最简化的结果，在本例中的结果是 1/(x-1)/x。

step 3 简化显示结果。如果只想查看最后的结果，则需要将化简的表达式赋给变量，在命令窗口中输入下列表达式，得到的结果如下：

```
>> R=simple(f);
>> R
 R =
```

```
1/(x-1)/x
```

可以看出，当将simple的结果赋给变量R后，MATLAB就只会显示最简化的结果，而省去了所有的中间结果。

根据基础数学知识，可以将表达式化简如下：

$$\sqrt{\frac{1}{x^2}+\frac{1}{(x-1)^2}-2\times\frac{1}{x(x-1)}}=\sqrt{(\frac{1}{x}-\frac{1}{x-1})^2}=\frac{1}{x-1}-\frac{1}{x}=\frac{1}{x(x-1)}$$

对比上面的结果，MATLAB使用simple命令得到的简化结果和基础数学知识化简的结果相同。

例5.18 在MATLAB中使用simple命令化简表达式 $\cos\theta+\sqrt{-(\sin\theta)^2}$。

step 1 在命令窗口输入以下内容：

```
>> syms theta
>> p=cos(theta)+sqrt(-sin(theta)^2);
>> R1=simple(p);
>> R2=simple(R1);
```

step 2 在MATLAB的命令窗口中输入"R=[R1;R2]"，然后按"Enter"键，得到的结果如下：

```
>> R=[ R1;R2]
 R =

 cos(theta)+i*sin(theta)
           exp(i*theta)
```

从上面的程序代码中可以看出，可以多次使用simple命令来达到最简的表达式，上面的表达式R1和R2都是化简的步骤：

$$\cos\theta+\sqrt{-(\sin\theta)^2}=\cos\theta+i\sin\theta=e^{i\theta}$$



尽管simple和simplify命令都是用来化简表达式的，但是simple命令的功能比较强大。上面的例子表明simple命令可以多次简化表达式，而simplify命令则不能多次简化表达式。

5.3.8 Pretty 函数

在MATLAB中，pretty函数的功能是用习惯的"书写"方式来显示符号表达式，其常见的调用格式如下：

- pretty(S) 将符号表达式用书写方式显示出来，使用默认的宽度79
- pretty(S,n) 将符号表达式用书写方式显示出来，使用指定的宽度n

例5.19 在MATLAB中使用pretty函数显示数值矩阵。

step 1 在MATLAB的命令窗口中输入下面的内容，创建符号对象：

```
>> A = sym(Pascal(3));
>> B=eig(A);
```

step 2 查看并简化符号矩阵B，在命令窗口中输入下面的内容：

```
>> B
 B =

              1
 4+15^(1/2)
 4-15^(1/2)
 >> pretty(B)
```

```
[      1     ]
[            ]
[        1/2]
[4 + 15     ]
[            ]
[        1/2]
[4 - 15     ]..
```



在上面的程序代码中，Pascal命令返回的是数值矩阵，该函数是MATLAB中的内置函数。而后面的程序代码中分别显示B和书写形式的矩阵B。

例5.20 在MATLAB中使用pretty函数显示符号表达式。

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> syms x
P=(x^2-1)/(x+2)+(2*x+5)/(3*x-2);
```

step 2 查看书写形式的符号表达式P，在命令窗口中输入下面的内容：

```
>> pretty(P)
```

```
  2
x~  - 1   2 x~ + 5
------- + --------
x~ + 2    3 x~ - 2
```

从上面的结果中可以看出，pretty函数除了可以简化符号数值矩阵之外，还可以简化符号表达式。

例5.21 在MATLAB中使用pretty函数显示一元二次方程的通解。

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> syms a b c x
>> f=solve('a*x^2+b*x+c');
```

step 2 查看书写形式的符号表达式f，在命令窗口中输入下面的内容：

```
>> pretty(f)
```

```
[               2            1/2]
[       -b + (b  - 4 a c)       ]
[1/2 ---------------------------]
[                a              ]
[                               ]
```

```
[                2             1/2]
[       -b - (b   - 4 a c)        ]
[ 1/2 -------------------]
[                a                 ]
```

 符号表达式的替换

在MATLAB中，符号计算的结果一般比较复杂，显得比数值计算烦琐，其中一个主要原因是有些子表达式多次出现在不同的地方。为了解决这个问题，MATLAB提供通过符号替换的方式来使表达式的输出形式简化，进而得到比较简单的表达式。

在MATLAB的符号工具箱中，提供两个函数subexpr和subs，实现符号表达式的替换，下面分小节详细介绍这两个命令。

 Subexpr 函数

在MATLAB中，subexpr函数的功能是将表达式中重复出现的字符串用变量替换，其常用的调用格式如下。

- [Y,SIGMA] = subexpr(X,SIGMA) 指定用变量SIGMA的值（该变量必须是符号对象）来替代符号表达式中重复出现的字符串。替换的结果由变量Y返回，被替换的字符串则由变量SIGMA代替。
- [Y,SIGMA] = subexpr(X,'SIGMA') 这种形式和上一种形式的区别在于，第二输入参数是字符或者字符串，它用来替换符号表达式中重复出现的字符串。

例 5.22 在MATLAB中，使用相关命令来求解方程 $ax^3+bx^2+cx+d=0$ 的通解表达式。

step 1 在MATLAB 的命令窗口中输入下面的内容：

```
>> syms a b c d x
>> t = solve('a*x^3+b*x^2+c*x+d = 0');
```

step 2 在MATLAB的命令窗口中输入"R=subexpr(t)"，然后按"Enter"键，得到系统默认的化简结果如下：

```
>> R=subexpr(t)
 sigma =

36*c*b*a-108*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-
18*c*b*a*d+27*d^2*a^2+4*d*b^3)^(1/2)*a

R =
1/6/a*sigma^(1/3)-2/3*(3*c*a-b^2)/a/sigma^(1/3)-1/3*b/a
 -1/12/a*sigma^(1/3)+1/3*(3*c*a-b^2)/a/sigma^(1/3)-1/3*b/a
+1/2*i*3^(1/2)*(1/6/a*sigma^(1/3)+2/3*(3*c*a-b^2)/a/sigma^(1/3))
 -1/12/a*sigma^(1/3)+1/3*(3*c*a-b^2)/a/sigma^(1/3)-1/3*b/a
-1/2*i*3^(1/2)*(1/6/a*sigma^(1/3)+2/3*(3*c*a-b^2)/a/sigma^(1/3))
```

step 3 接着在MATLAB 的命令窗口中输入下面的内容：

```
[r,s] = subexpr(t,'s');
```

step 4 然后在 MATLAB 的命令窗口中依次输入 s、r，查看化简结果如下：

```
>> s
 s =

36*c*b*a-108*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-18*c*b*a*d+27*d^2*a^2+4*d*b^3)^(1/2)*a
 >> r
 r =
1/6/a*s^(1/3)-2/3*(3*c*a-b^2)/a/s^(1/3)-1/3*b/a
 -1/12/a*s^(1/3)+1/3*(3*c*a-b^2)/a/s^(1/3)-1/3*b/a+1/2*i*3^(1/2)*(1/6/a*s^(1/3)+2/3*(3*c*a-b^2)/a/s^(1/3))
 -1/12/a*s^(1/3)+1/3*(3*c*a-b^2)/a/s^(1/3)-1/3*b/a-1/2*i*3^(1/2)*(1/6/a*s^(1/3)+2/3*(3*c*a-b^2)/a/s^(1/3))
```

从上面可以看出，当没有输入替换参数的时候，MATLAB 会使用默认值 SIGMA。上面的计算结果很烦琐，但是使用 simple 命令并不能简化，因为表达式中并不存在可以合并、展开或者因式分解的表达式，但是 subexpr 命令使用替换的方法就可使整个表达式在外观上显得简化了很多。



在MATLAB中，符号表达式中被置换的子表达式是系统自行寻找的，其置换的原理和 pretty 命令相似。在 pretty 命令中，使用%1、%2 替换比较长的子表达式。而且，只有比较长的子表达式才会被置换，对于比较短的表达式，即使存在多次重复的子表达式，也不会被替换。

5.4.2 Subs 函数

在 MATLAB 中，subs 函数的功能是使用指定符号替换符号表达式中的某一个特定符号，相对于 subexpr 命令，subs 命令是一个通用的替换命令，subs 命令的常用调用格式如下。

- R = subs(S) 用工作空间中的变量替换符号表达式 S 中的所有符号变量，如果没有指定某符号变量的值，则返回值中该符号变量不被替换。
- R = subs(S, new) 用新的符号变量 new 来替换原来符号表达式 S 中的默认变量。确定默认变量的规则和函数 findsym 规则相同。
- R = subs(S,old,new) 用新的符号变量 new 替换原来符号表达式 S 中的变量 old，当 new 是数值形式的符号时，实际上用数值替换原来的符号来计算表达式的值，只是所得结果还是字符串形式。

例 5.23 在 MATLAB 中，使用 subs 函数替换符号表达式中的变量。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> clear all
a=980;C1=3;
syms y
y = dsolve('Dy = -a*y');
```

step 2 使用 subs 函数查看默认的替换结果：

```
>> [y;subs(y)]
 ans =

  C1*exp(-a*t)
 3*exp(-980*t)
```

step 3 在 MATLAB 的命令窗口中输入下面的内容：

```
>> syms alpha theta t
>> P=subs(exp(theta*t),'theta',-magic(2));
```

step 4 查看上面的替换结果：

```
>> P
 P =

[    exp(-t), exp(-3*t)]
[  exp(-4*t), exp(-2*t)]
```

step 5 在 MATLAB 的命令窗口中输入下面的内容：

```
>> syms theta alpha gamma delta
>> R=subs(cos(theta)+sin(alpha),{theta,alpha},{gamma,delta});
```

step 6 查看上面的替换结果：

```
>> R
 R =

cos(gamma)+sin(delta)
```



上面的例子并不复杂，但是演示了subs函数一些常见用法，可以选择类似的例子来尝试各种不同的使用方法和结果。

5.5 符号函数的操作

在 MATLAB 中，在处理符号对象的时候，除了需要了解前面小节中介绍的符号表达式内容之外，还需要了解关于符号函数的相应操作。在MATLAB7.0的符号工具箱中，提供了关于符号函数的求反和复合函数，下面详细介绍这两类函数的应用。

5.5.1 Finverse 函数

在 MATLAB 中，finverse 函数的功能是求某函数的反函数。反函数是高等数学的一个基本内容，其基本概念如下：对于函数 $f(x)$，在实数范围内，存在某函数 $g(g)$，使得 $g(f(x))=x$，则函数 $g(g)$就被称为函数 $f(x)$的反函数。

finverse 命令的常见调用格式如下：

- g = finverse(f) 对默认的自变量求反函数 g；
- g = finverse(f,v) 对指定的自变量为 v 的 f(v)，求反函数 g(v)。

例 5.24 在 MATLAB 中，分别求解不同类型函数的反函数。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> syms t x
>> f1=finverse(log(t));
>>f2=finverse(sin(2*t));
>> f3=finverse(exp(t-2*x),x);
>>f4=finverse(exp(t-2*x),t);
>> g=[ f1;f2;f3;f4] ;
```

step 2 在 MATLAB 的命令窗口中输入"g"，然后按"Enter"键，得到的结果如下：

```
>> g
 g =

            exp(t)
       1/2*asin(t)
 1/2*t-1/2*log(x)
        2*x+log(t)
```



在上面的程序代码中，分别使用 finverse 函数求得了对数、三角函数和多元函数的反函数。其中，当对多元函数指定不同的变量时，求得的反函数则会不同。

例 5.25 在 MATLAB 中，求解函数 $f(t)=t^2$ 的反函数。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> syms t
>> f1=t^2;
>> g1=finverse(f1)
```

step 2 在输入上面的代码后，按"Enter"键，得到下面的结果：

```
Warning: finverse(t^2) is not unique.
> In sym.finverse at 43
 g1 =
 t^(1/2)
```

当某个函数的反函数不唯一的时候，MATLAB会出现上面的提示信息："Warning: finverse(t^2) is not unique."该信息提示用户使用 finverse 返回的只是多个反函数中的一个，在本实例中得到的结果是 $\sqrt{t}$，而没有显示另外一个反函数 $-\sqrt{t}$。

5.5.2 Compose 函数

在MATLAB中，compose函数的功能是产生复合函数。复合函数是数学分析中经常遇到的问题，其基本概念为：对于函数 $f(x)$ 和 $x=g(y)$，两个函数的复合函数就是 $f(g(y))$。

在 MATLAB 中，compose 命令的常用调用格式如下：

- compose(f,g) 对 $f(g)$ 和 $u=g(g)$ 求得复合函数 $fg=f(g(g))$。
- compose(f,g,x,y,z) 对函数 $f(x)$ 和 $v=g(y)$，求得复合函数 $fg=f(g(y))|_{y=z}$。

例 5.26 在 MATLAB 中，实现复合函数。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> syms x y z t u;
>> f = 1/(1 + x^2);
>> g = sin(y);
>> h = x^t;
>> p = exp(-y/u);
>> fgy=compose(f,g);
fgt=compose(f,g,t);
hgt=compose(h,g,x,z);
hgx=compose(h,g,t,z);
hpt=compose(h,p,x,y,z);
hpx=compose(h,p,t,u,z);
>> comp=[ fgy;fgt;hgt;hgx;hpt;hpx] ;
```

step 2 在 MATLAB 的命令窗口中输入 "comp"，然后按 "Enter" 键，查看使用复合函数得到的结果如下：

```
>> comp
 comp =

 1/(1+sin(y)^2)
 1/(1+sin(t)^2)
        sin(z)^t
        x^sin(z)
    exp(-z/u)^t
x^exp(-y/z)
```

从上面的结果中可以看出，对于相同的两个基础函数，如果使用不同类型的变量，复合函数的结果会互不相同，因此，在进行函数复合时，需要注意变量的选择。



在 MATLAB 中，compose(f,g)命令其实是 compose(f,g,x,y,z)命令的默认形式。在这种情况下，x、y 都是由 findsym 函数自动对 f 和 g 函数进行辨认而定的自变量。

5.6 符号微积分

在数学分析中，微积分一直是一个十分主要的内容，整个高等数学就是建立在微积分运算的基础上的，同时，微积分也是微分方程体系的基础内容。在 MATLAB 中，提供一些常见的函数来支持这些微分运算，所涉及到的领域包括极限、微分、积分和级数等各个方面。

尽管和数值计算相比，符号运算一般需要消耗更多的内存资源，但是这并不意味着符号运算没有使用的场合。在有些情况下，符号计算处理比数值计算会更加便捷。下面将分小节详细介绍符号运算在微积分中的应用。



diff 函数

当创建符号表达式后，就可以使用diff函数来对它们进行微分运算。在MATLAB中，diff命令的常用调用格式如下：

- diff(S,'v') 将符号'v'当作变量，对符号表达式或者符号矩阵S求得微分；
- diff(S,n) 将符号表达式S中的默认变量进行n阶微分预算，其中默认的变量可以使用函数findsym来确定，参数n必须是正整数；
- diff(S,'v',n) 将符号'v'当作变量，对符号表达式或者符号矩阵S求得n阶微分。

例5.27 在MATLAB中，使用diff函数来求解$f(x,y)=x^2+2xy-3y^2$的一阶和二阶偏导数值。

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> syms x y
>> f=x^2+2*x*y-3*y^2;
>>dfdx=diff(f,x);
>>dfdy=diff(f,y);
>>dfdxdy=diff(dfdx,y);
>>dfdydx=diff(dfdy,x);
```

step 2 在MATLAB的命令窗口中输入"[dfdx;dfdy;dfdxdy;dfdydx]"，然后按"Enter"键，可以得到下面的结果：

```
>> [ dfdx;dfdy;dfdxdy;dfdydx]

 ans =

 2*x+2*y
 2*x-6*y
       2
       2
```

从上面的程序代码中，通过使用简单的diff命令，就可以证明微积分中的一个结论：$\frac{\partial^2 f(x,y)}{\partial x\partial y}=\frac{\partial^2 f(x,y)}{\partial y\partial x}$。在本例中，$f(x,y)=x^2+2xy-3y^2$，因此根据上面程序的结果，两者满足$\frac{\partial^2 f(x,y)}{\partial x\partial y}=\frac{\partial^2 f(x,y)}{\partial y\partial x}=2$。



化简微分结果

例5.28 在MATLAB中，使用diff函数来求解常见的多项式$\ln(x+\sqrt{x^2\pm a^2})$的一阶导数数值，并使用MATLAB中相关的命令简化求解的结果。

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> syms x a
>> f1=diff(log(x+sqrt(x^2+a^2)));
>> f2=diff(log(x+sqrt(x^2-a^2)));
```

step 2 在MATLAB的命令窗口中依次输入f1、f2，查看求导结果如下：

```
>> f1
 f1 =

(1+1/(x^2+a^2)^(1/2)*x)/(x+(x^2+a^2)^(1/2))
>> f2
 f2 =

(1+1/(x^2-a^2)^(1/2)*x)/(x+(x^2-a^2)^(1/2))
```

step 3 对上面的结果进行化简，使用前面介绍的simple命令，得到的结果如下：

```
>> D1=simple(f1);
>> D2=simple(f2);
>> D1

 D1 =

1/(x^2+a^2)^(1/2)
>> D2
 D2 =

1/(x^2-a^2)^(1/2)
```

根据相关的微积分知识，多项式$\ln(x+\sqrt{x^2\pm a^2})$的微分结果如下：

$\frac{d}{dx}\ln(x+\sqrt{x^2\pm a^2})=\frac{1}{\sqrt{x^2\pm a^2}}$。



从上面的例子可以看出，diff函数的结果并不一定是最简洁的，如果希望得到最简的微分结果，经常需要使用前面介绍的多种化简命令。

5.6.3 求解矩阵微分

例5.29 在MATLAB中，求解多项式矩阵$A=\begin{pmatrix} s\times\cos t & t^2\ln s \\ 2^s\times\sin(2t) & s^3\ln(2+t) \end{pmatrix}$的下面各阶微分数值：$\frac{dA}{dt}$、$\frac{d^2A}{ds^2}$和$\frac{d^2A}{dsdt}$。

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> syms s t
>> f=[ s*cos(t),t^2*log(s);2^s*sin(2*t),s^3*log(2+t)];
>> df=diff(f);
>> dfds2=diff(f,s,2);
>> dfdsdt=diff(diff(f,t),s);
```

step 2 在MATLAB的命令窗口中依次输入上面的各个求导变量，查看求导结果如下：

```
>> df
 df =

[        -s*sin(t),      2*t*log(s)]
[  2*2^s*cos(2*t),       s^3/(2+t)]
```

```
>> dfds2
 dfds2 =

[                    0,              -t^2/s^2]
[  2^s*log(2)^2*sin(2*t),        6*s*log(2+t)]
>> dfdsdt
 dfdsdt =

[              -sin(t),                  2*t/s]
[  2*2^s*log(2)*cos(2*t),         3*s^2/(2+t)]
```



从上面的例子中可以看出，diff命令不仅可以适用于单个多项式、变量，还可以适用于多项式矩阵，灵活使用diff命令可以求解各种微分数值。

5.6.4 向量微分 Jacobian 函数

在数学分析中，微分运算也可以对列向量进行，所得的结果也是一个列向量。在数学分析中，多元向量函数f的jacobian矩阵的定义如下：

对于多元向量函数 $f(v)=\begin{bmatrix} f_1(v) \\ \vdots \\ f_n(v) \end{bmatrix}$ 和向量变量 $v=[v_1,\cdots,v_m]$，其函数f的jacobian矩阵为

$$f(v)=\begin{bmatrix} \frac{\partial f_1}{\partial v_1} & \cdots & \frac{\partial f_1}{\partial v_m} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_n}{\partial v_1} & \cdots & \frac{\partial f_n}{\partial v_m} \end{bmatrix}。$$



当f是矩阵的时候，则求导必须对各个元素进行，但是自变量必须定义在整个矩阵上。

在MATLAB中，jacobian的调用格式如下：

R = jacobian(f,v)　其中f是一个符号列向量，v是指定进行变换的变量组成的行向量。

例 5.30　沿用例5.29中的矩阵$\boldsymbol{A}$，将其转换为两个列向量$\boldsymbol{B}_1$和$\boldsymbol{B}_2$，分别求解两个列向量的jacobian矩阵。

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> syms s t
f=[ s*cos(t),t^2*log(s);2^s*sin(2*t),s^3*log(2+t)];
>> g1=f(:,1);
>> g2=f(:,2);
>> v=[ s t];
>> g1jacob=jacobian(g1,v);
>> g2jacob=jacobian(g2,v);
```

step 2 在MATLAB的命令窗口中输入"g1jacob、g2jacob"，查看列向量的jacobian矩阵：

```
>> g1jacob
```

```
g1jacob =

[            cos(t),         -s*sin(t)]
[ 2^s*log(2)*sin(2*t),      2*2^s*cos(2*t)]
>> g2jacob
g2jacob =

[          t^2/s,       2*t*log(s)]
[ 3*s^2*log(2+t),       s^3/(2+t)]
```

根据例5.29可以得知，$A=\begin{pmatrix} s\times\cos t & t^2\ln s \\ 2^t\times\sin(2t) & s^3\ln(2+t)\end{pmatrix}$，在本例中可以从中得到的列向量为$B_1=\begin{pmatrix} s\times\cos t \\ 2^t\times\sin(2t)\end{pmatrix}$和$B_2=\begin{pmatrix} t^2\ln s \\ s^3\ln(2+t)\end{pmatrix}$。然后分别求解上面两个列向量的jacobian矩阵。

step 3 求解两个jacobian矩阵的行列式，得到的结果如下：

```
>> detJ1=simple(det(g1jacob));
>> detJ2=simple(det(g2jacob));
>> [detJ1;detJ2]

ans =

      2^s*(2*cos(t)*cos(2*t)+s*sin(t)*log(2)*sin(2*t))
 (t-12*log(s)*log(2+t)-6*log(s)*log(2+t)*t)*t*s^2/(2+t)
```



雅可比函数的第一个参数是列向量，而第二个参数则必须是行向量。同时，雅可比矩阵的行列式一般比较复杂，需要使用符号表达式的简化命令来进行化简。

5.6.5 符号极限

根据高等数学的基础知识，表达式的极限是微分的基础。极限的定义则是当自变量趋近某个范围或者数值时，函数表达式的数值。无穷逼近也是微积分的基础思想，因此极限是整个微积分的基础。

在MATLAB中，提供函数limit求解表达式或者函数的极限，其常用的调用格式如下：

- limit(F,x,a) 求解当x→a时，符号表达式F的极限；
- limit(F,a) 符号表达式F采用默认自变量，该函数求得F的自变量趋近于a时的极限值；
- limit(F) 符号表达式F采用默认自变量，并且以a=0为自变量的趋近值，该函数求得F的自变量趋近于a时的极限值；
- limit(F,x,a,'right') 该函数求解符号表达式F右极限，也就是自变量从右边趋近于a的函数极限值；
- limit(F,x,a,'left') 该函数求解表达式F的左极限，也就是自变量从左边趋近于a的函数极限值。

例5.31 在MATLAB中，求解表达式$\lim\limits_{x\to0}\frac{\tan x-\sin x}{x^3}$和$\lim\limits_{x\to0}\frac{\tan x-x}{x^3}$的极限数值。

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> syms x
>> F1=limit((tan(x)-sin(x))/x^3);
>> F2=limit((tan(x)-x)/x^3);
```

step 2 在命令窗口中输入"[F1,F2]"，查看极限表达式的数值如下：

```
>> [F1,F2]

ans =

[ 1/2, 1/3]
```

根据微积分的相关知识，求解表达式的极限可以使用各种方法，在MATLAB中程序会调用各种方法来求解各种极限数值。



上面的实例中使用的是MATLAB的默认格式来求解极限数值。MATLAB会自动认定自变量x，同时认定自变量x的趋近数值为0。

5.6.6 求解无限极限

例 5.32 在MATLAB中，证明常用极限表达式$\lim\limits_{n\to 0}(1\pm\frac{x}{n})^n=e^{\pm x}$。

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> syms x
>> F1=limit( (1 + x/n)^n,n,inf );
>> F2=limit( (1 - x/n)^n,n,inf );
```

step 2 在命令窗口中输入"[F1;F2]"，查看极限表达式的数值如下：

```
>> [F1;F2]
 ans =
  exp(x)
 exp(-x)]
```



根据limit命令的格式，当极限的范围是无穷大时，MATLAB中用字符inf来替代，得到的结果证明MATLAB的相关命令正确。

5.6.7 求解左右极限

例 5.33 在MATLAB中，求解$f(x)$的在x=0处的左右极限和极限数值，其中函数$f(x)$当$x\neq 0$时，$f(x)=\frac{x}{|x|}$；当x=0时，$f(x)$=0。

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> syms x
>> fl=limit(x/abs(x),x,0,'left');
>> fr=limit(x/abs(x),x,0,'right');
>> f=limit(x/abs(x),x,0);
```

step 2 在命令窗口中输入 "[fl;fr;f]"，查看极限表达式的数值如下：

```
>> [ fl;fr;f]
 ans =

   -1
    1
  NaN
```



从上面的求解结果可以看出，左极限的数值为 -1，右极限的数值为 1。由于左右极限数值不等，则可以得出结果极限不存在，在 MATLAB 中用 NaN 来表示。

step 3 绘制函数图形。最后，为了了解上面表达式的极限情况，可以绘制该函数的图形来形象地了解函数性态，相应的程序代码如下：

```
>> x1=-1:0.001:0;
y1=x1/abs(x1);
x2=0:0.001:1;
y2=x2/abs(x2);
plot(x1,y1,'r',x2,y2,'r')
axis([ -1 1 -1.4 1.4])
title('x/|x|')
```

得到的图形如图 5.1 所示。

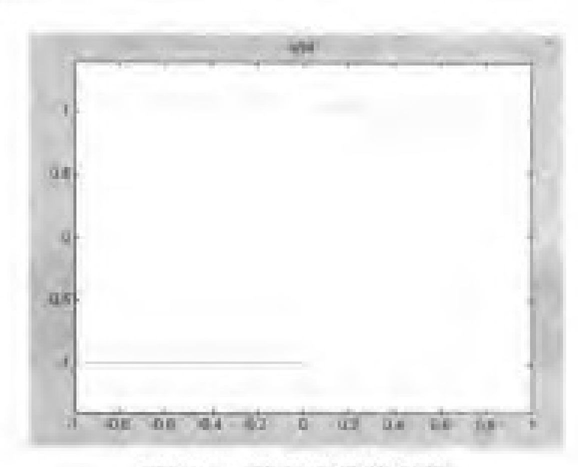


图 5.1　绘制函数的图形

从上面的图形中，可以很清楚地看到，函数在 x=0 的地方间断，左侧为数值 -1，右侧为数值 1，因此函数在 x=0 的地方不存在极限数值。



上面的程序代码都是用来绘制函数图形的。如果读者不太熟悉相关的命令，可以参考本书中介绍 MATLAB 绘图的章节。

5.6.8 符号积分

在数学分析中，积分和微分是一种互逆的运算。积分包括不定积分、定积分、旁义积分和重积分等。一般来讲，积分比微分更难求解。在MATLAB的符号数学工具箱中，提供函数int来求解符号积分。int命令可以直接接通MAPLE，进行十分有效的求积。

和数值积分相比，符号积分的指令简单，适应性比较强，但是可能会占用较长的时间。有时符号积分可能会给出比较冗长的符号表达式。如果求解的是不可积的表达式，int命令会返回积分的原式并显示警告信息。

在MATLAB中，int命令的常用调用格式如下。

- R = int(S) 用默认的变量求符号表达式S的不定积分。
- R = int(S,v) 用符号变量v作为变量求符号表达式S的不定积分数值。
- R = int(S,a,b) 符号表达式采用默认变量，该函数求默认变量从a变到b时符号表达式S的定积分数值。如果S是符号矩阵，则积分对各个元素分别进行积分。
- R = int(S,v,a,b) 用符号变量v作为变量求符号表达式S的定积分数值。



在int命令中，a和b不仅可以是常数变量，也可以是符号表达式和其他数值表达式，分别表示积分表达式的上限和下限。

例5.34 在MATLAB中，求表达式$\frac{1}{\sqrt{x^2 \pm a^2}}$的不定积分结果。

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> syms x a
>> I1=int(1/sqrt(x^2+a^2));
>> F1=simple(I1);
>> I2=int(1/sqrt(x^2-a^2));
>> F2=simple(I2);
>> F=[ F1;F2] ;
```

step 2 在命令窗口中输入"F"，然后按"Enter"键，查看不定积分的结果如下：

```
>> F
F =

 log(x+(x^2+a^2)^(1/2))
 log(x+(x^2-a^2)^(1/2))
```

根据上面的结果可以看出，积分结果满足下列的积分等式：

$$\int \frac{1}{\sqrt{x^2 \pm a^2}} \mathrm{d}x = \ln(x+\sqrt{x^2 \pm a^2})+C$$

MATLAB给出了其中的一个不定积分结果$\ln(x+\sqrt{x^2 \pm a^2})$，没有给出其中的常数C。



从上面的程序可以看出，int命令得到的结果一般比较冗长，需要使用简化命令来简化结果。

5.6.9 矩阵积分

例 5.35 在 MATLAB 中，求矩阵$A=\begin{pmatrix}\cos t & t^2 \\ 2^t & \ln(2+t)\end{pmatrix}$的积分结果。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> syms t
>> f=[ cos(t),t^2;2^t,log(2+t)] ;
>> I=int(f);
```

step 2 在命令窗口中输入"I"，然后按"Enter"键，查看不定积分的结果如下：

```
>> I
 I =

[              sin(t),              1/3*t^3]
[        1/log(2)*2^t, log(2+t)*(2+t)-2-t]
```

step 3 在命令窗口中输入"pretty(I)"，然后按"Enter"键，查看新的结果如下：

```
>> pretty(I)

                       [                         3           ]
                       [ sin(t)              1/3 t           ]
                       [                                     ]
                       [    t                                ]
                       [   2                                 ]
                       [ ------    log(2 + t) (2 + t) - 2 - t]
                       [ log(2)                              ]
```



从上面的分析结果中可以看出，当积分对象是符号矩阵的时候，int命令会对矩阵中的元素依次进行积分，得出积分结果。

5.6.10 证明积分等式

例 5.36 在 MATLAB 中，使用 int 积分命令验证正态分布的结果。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> syms a positive
>> syms x;
>>f = exp(-a*x^2);
>>R=int(f,x,-inf,inf);
```

step 2 在命令窗口中输入"R"，然后按"Enter"键，查看定积分的结果如下：

```
>> R

R =
```

```
1/a^(1/2)*pi^(1/2)
```

step 3 在 MATLAB 的命令窗口中输入下面的内容：

```
>> syms x
>> syms a real
>>f=exp(-a*x^2);
>>F = int(f, x, -inf, inf);
```

step 4 在命令窗口中输入"F"，然后按"Enter"键，查看定积分的结果如下：

```
>> F
 F =

PIECEWISE([ 1/a^(1/2)*pi^(1/2), signum(a) = 1],[ Inf, otherwise])
```

step 5 查看简化结果。在命令窗口中输入"pretty(F)"，然后按"Enter"键，查看简化后的定积分结果：

```
>> pretty(F)
                    {    1/2
                    { pi
                    { -----          signum(a~) = 1
                    {    1/2
                    { a~
                    {
                    {  Inf              otherwise
```

根据微积分的相关知识，标准正态分布满足 $\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} e^{\frac{-(x-u)^2}{2\delta^2}}dx=1$。因此，在本实例中，使用相应的积分变换可以得到 $\int_{-\infty}^{+\infty} e^{-ax^2}dx=\sqrt{\frac{\pi}{a}}$。在上面的程序代码中，得出的结果就是1/a^(1/2)*pi^(1/2)。在后面的步骤中，将a设定为任意实数的符号变量，MATLAB将会根据变量a的不同情况来得到积分结果。



在上面的程序代码的后面步骤中，将a设定为real，这时候变量a既是MATLAB工作空间的变量，也是MAPLE内核工作空间的一个实变量。如果用户希望清除该变量a，使用命令clear a只能将其在MATLAB的工作空间中删除该符号对象，在MAPLE的内核工作空间中则存在着变量，应该使用"syms a unreal"将其从MAPLE的内核中清除。

5.6.11 交互近似积分

除了上面提供的int命令之外，MATLAB还提供一个交互性的近似积分命令rsums，该命令可以计算一元函数在某有限的闭区间上的积分数值。在MATLAB中，rsums命令的调用格式为：rsums(f,a,b)，其中f是积分表达式，a和b分别为积分的上下限。

例 5.37 在 MATLAB 中，使用 rsums 命令求解函数 $f(x)=(x-1)^3+x^2+4x$ 在积分区间的积分结果。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> syms x;
>> f=(x-1)^3+x^2+4*x;
>> rsums(f,-1,2)
```

step 2 在命令窗口输入上面的命令后，按"Enter"键，MATLAB 会自动调用近似积分的交互界面，如图 5.2 所示。

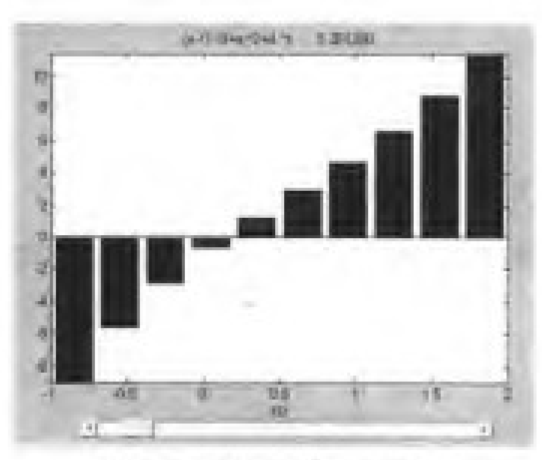


图 5.2　交互近似积分界面



在默认的情况下，交互近似积分界面的下方有一个"滑动键"，该滑动键用来设置函数曲线下方的矩形个数。默认情况下，矩形的个数为 10。当"滑动键"向右调节时，矩形数会增加，最大的矩形个数为 128。

step 3 调整积分矩形的个数。向右调整"滑动键"，将其设置为 128，查看近似积分的数值，如图 5.3 所示。

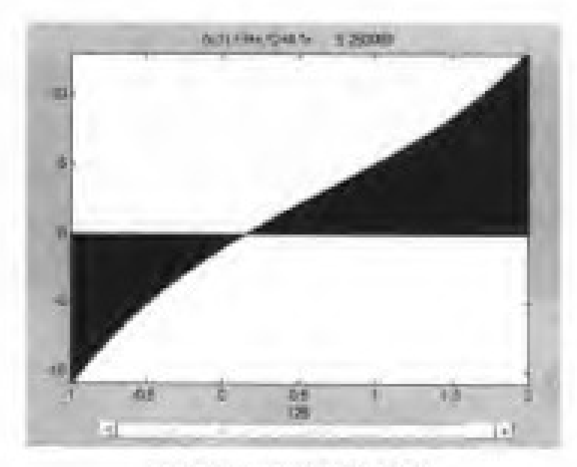


图 5.3　调整积分精度

step 4 在命令窗口中输入"int(f,-1,2)"，计算函数的准确积分数值，结果如下：

```
>> int(f,-1,2)
 ans =
```

```
21/4
```



从上面的结果中可以看出，精确值为21/4=5.25，而上面近似积分的数值为5.250069，精度应该可以达到一般的要求。

5.6.12 符号级数求和

在数学分析中，级数求和是一个重要的内容，MATLAB提供symsum命令来对解符号表达式进行求和。symsum 命令的调用格式如下：

- r = symsum(s,a,b) 将符号表达式 s 中的默认变量从 a 变到 b 时的有限和；
- r = symsum(s,v,a,b) 将符号表达式 s 中的变量 v 从 a 变到 b 时的有限和。

例 5.38 在 MATLAB 中，使用 symsum 命令来求解各种常见级数求和。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> syms x k
>>s1=symsum(1/k^2,1,inf);
>> s2=symsum(k^2);
>> s3=symsum(x^k/sym('k!'), k, 0,inf) ;
```

step 2 在命令窗口中输入 "S=[s1;s2;s3]"，然后按 "Enter" 键，查看求解的结果如下：

```
>> S=[ s1;s2;s3]
 S =

                 1/6*pi^2
 1/3*k^3-1/2*k^2+1/6*k
                   exp(x)
```

上面的计算结果分别对应的级数运算如下：

$$\sum_{k=1}^{\infty}\frac{1}{k^2}=\frac{\pi^2}{6}$$

$$\sum_{k=1}^{\infty}k^2=\frac{2k^3-3k^2+k}{6}=\frac{k(k-1)(2k-1)}{6}$$

$$\sum_{k=0}^{\infty}\frac{x^k}{k!}=e^x$$

关于上面各种级数运算的结果，可以查看相应的数学书籍。



在上面的表达式s3中，用sym命令将 "k!" 转换为符号变量，在后来的运算过程中，MATLAB会将 "k!" 视为符号。由于MATLAB中没有定义阶乘的运算法则，如果不使用该命令，MATLAB会返回错误信息。

5.7 符号积分变换

在数学分析中，通过数学变换将复杂的计算转换为简单的计算是一个重要的手段。其中，积分变换是数学变换中的一个重要内容。所谓积分变换，就是通过积分计算，把一类函数 A 变换成另一类函数 B，函数 B 一般是含有参量 a 的积分 $\int_a^b f(t)k(t,a)\mathrm{d}t$。这一变换的目的就是，将函数类 A 中的函数 $f(t)$ 通过积分运算变成另外一个函数类 B 中的函数 $f(a)$。积分变换计算公式中的 $k(t,a)$ 就是积分变换的核，而 $f(t)$ 叫做原函数，$f(a)$ 叫做像函数。

积分变换的方法在自然科学和工程实际中有着广泛的应用，是一个不可或缺的运算工具。在本节中，将介绍三个主要的积分变换：Fourier 变换、Laplace 变换和 Z 变换。

5.7.1 Fourier 变换命令

在时域中的 $f(t)$ 与它在频域中的 Fourier 变换 $F(w)$ 之间存在着如下的关系：

$$F(w)=\int_{-\infty}^{\infty} f(t)e^{-jwt}\mathrm{d}t$$

$$f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty} F(w)e^{jwt}\mathrm{d}w$$

根据上面的积分定义，可以使用前面介绍的 int 命令直接求解，但是，MATLAB 中提供专门的 Fourier 变换程序：Fourier 和 ifourier 命令。在本小节中，将详细介绍 MATLAB 中的这些命令，对于积分定义的方法，感兴趣的读者请自行尝试。

在 MATLAB 中，Fourier 和 ifourier 命令的调用格式如下：

- Fw=fourier(ft,t,w)　求时域上函数 ft 的 Fourier 变换 Fw，其中 ft 是以 t 为自变量的时域函数，Fw 是以圆频率 w 为自变量的频域函数；
- ft=ifourier(Fw,w,t)　求频域上函数 Fw 的 Fourier 反变换 ft，其中 ft 是以 t 为自变量的时域函数，Fw 是以圆频率 w 为自变量的频域函数。

5.7.2 Fourier 变换实例

例 5.39　在 MATLAB 中，分别使用 fourier 命令和原始积分定义求解表达式的积分结果。

step 1 在 MATLAB 的命令窗口输入下面的内容：

```
>> syms x w t
>>f1=exp(-x^2);
>>g1=f*exp(-i*w*x);
>>fy1=fourier(f1);
>>fd1=int(g1,x,-inf,inf);
>> f2=exp(-abs(w));
>>g2=f2*exp(-i*w*t);
>> fy2=fourier(f2);
>>fd2=int(g2,w,-inf,inf);
```



在上面的程序代码中，fy1 和 fy2 是使用 fourier 变换命令求得的计算结果，fd1 和 fd2 则是根据积分定义计算得到的结果。读者可以比较两种不同计算方法得到的结果。

step 2 在 MATLAB 的命令窗口中依次输入计算结果的数组：

```
>> [ fy1;simple(fd1)]
 ans =
  pi^(1/2)*exp(-1/4*w^2)
 pi^(1/2)*exp(-1/4*w^2)
>> [ fy2;simple(fd2)]
 ans =
 2/(1+t^2)
 2/(1+t^2)
```

上面的程序代码中计算的积分表达式依次如下：

$f(x)=e^{-x^2}$，$F(w)=\int_{-\infty}^{\infty}f(x)e^{-iwx}dx$；根据积分运算，得到的结果是 $\sqrt{\pi}e^{-w^2/4}$；

$f(w)=e^{-|w|}$，$F(t)=\int_{-\infty}^{\infty}f(w)e^{-iwt}dw$；根据积分运算，得到的结果是 $\frac{2}{1+t^2}$。



根据上面的程序，使用 fourier 命令得到的结果和根据定义得到的结果一致，fourier 命令一般得到的就是比较简化的结果，而根据定义得到的结果则不一定比较简化。

例 5.40 使用 fourier 变换命令证明积分等式 $\int_{-\tau/2}^{\tau/2}Ae^{-iwt}dt=A\tau g\frac{\sin\frac{w\tau}{2}}{\frac{w\tau}{2}}$。

step 1 在 MATLAB 的命令窗口输入下面的内容：

```
>> syms A t w
>> syms tao positive
>> yt=sym('Heaviside(t+tao/2)-Heaviside(t-tao/2)');
>> Yw=fourier(A*yt,t,w);
```

step 2 在 MATLAB 的命令窗口中输入"Yw"，查看积分结果如下：

```
>> Yw
 Yw =

2*A/w*sin(1/2*tao*w)
```

由于积分表达式的上下限不是无限值，而是有限数值范围 $\left[-\frac{\tau}{2},\frac{\tau}{2}\right]$。为了能够使用 Fourier 变换求解积分表达式的数值，引入 Heaviside(t−a) 函数，该函数的数学含义是单位阶跃函数：当自变量 $t\geq a$ 时，函数的数值为 1；当自变量 $t<a$ 时，函数的数值为 0。

根据上面的分析，在上面的程序代码中的函数"Heaviside(t+tao/2)−Heaviside(t−tao/2)"的数值在自变量范围 $\left[-\frac{\tau}{2},\frac{\tau}{2}\right]$ 内为 1，在其他的范围内则为 0。这就等于变相限定了积分表达式的积分范围。

根据上面的程序代码，Fourier 变换得到的结果为 $\frac{2A}{w}g\sin\frac{w\tau}{2}=A\tau g\frac{\sin\frac{w\tau}{2}}{\frac{w\tau}{2}}$，因此上面的程序代码就等于证明了 $\int_{-\tau/2}^{\tau/2}Ae^{-iwt}dt=A\tau g\frac{\sin\frac{w\tau}{2}}{\frac{w\tau}{2}}$。



在MATLAB7.0的符号数学工具箱中，增加了dirac和heaviside两个函数，它们的功能是产生单位脉冲函数和阶跃函数。在以前的版本中，MATLAB只是借用了MAPLE函数库中关于heaviside函数的定义；如果希望在MATLAB中运算该函数时，必须从MAPLE函数库中调用。

5.7.3 Laplace 变换命令

在数学分析中，Laplace 变换和反变换的定义如下：

$$F(s)=\int_0^{\infty} f(t)e^{-st}\mathrm{d}t$$

$$f(t)=\frac{1}{2\pi j}\int_{C-j\infty}^{C+j\infty} F(s)e^{st}\mathrm{d}s$$

由于该变换也是用积分来定义的，因此，可以 int 命令直接求解 Laplace 变换。同时，MATLAB 中提供专门的 Laplace 变换程序：Laplace 和 ilaplace 命令。在本小节中，将详细介绍 MATLAB 中的这些命令；对于积分定义的方法，感兴趣的读者请自行尝试。

在 MATLAB 中，Laplace 和 ilaplace 命令的调用格式如下：

- **Fs=laplace(ft,t,s)**：求时域上函数 ft 的 Laplace 变换 Fs，其中 ft 是以 t 为自变量的时域函数，Fs 是以复频率 s 为自变量的频域函数；
- **ft=ilaplace(Fs,s,t)**：求频域上函数 Fs 的 Laplace 反变换 ft，其中 ft 是以 t 为自变量的时域函数，Fs 是以复频率 s 为自变量的频域函数。

5.7.4 Laplace 变换实例

例 5.41 证明 laplace 变换的时移性质 $L\{f(t-t_0)u(t-t_0)\}=e^{-st_0}L\{f(t)\}$。

其中，f 是任意一个函数，u 则是前面小节中介绍的阶跃函数，L 代表的是 Laplace 变换。

step 1 在 MATLAB 的命令窗口输入下面的内容：

```
>> syms t s
>> syms t0 positive
>> ft=Heaviside(t-t0)*sym('f(t-t0)');
>> FS=laplace(ft,t,s);
>> FS_t=ilaplace(FS,s,t);
```

step 2 在 MATLAB 的命令窗口中，查看上面的积分变化的结果：

```
>> ft
 ft =
 heaviside(t-t0)*f(t-t0)
>> FS
 FS =
 exp(-s*t0)*laplace(f(t),t,s)
>> FS_t
 FS_t =
 heaviside(t-t0)*f(t-t0)
```

从上面的程序代码中可以看出，$ft=f(t-t_0)u(t-t_0)$，FS 就是 ft 函数对应的 Laplace 变换的结果，

$L\{f(t-t_0)u(t-t_0)\}=e^{-st_0}L\{f(t)\}$。最后，FS_t 的结果是将FS 的反Laplace 变换，结果就是 $ft=f(t-t_0)u(t-t_0)$。



在本实例中，使用的Laplace变换和ilaplace变换都是完整的调用格式命令，这并不意味着laplace命令没有默认格式。但是，一般情况下，笔者建议读者不要使用默认格式，以免造成一些不必要的错误，同时还会占用计算机的内存资源。

5.7.5 Z变换命令

和前面两个变换不同，Z变换适用于离散的因果系列，Z变换和其反变换定义如下：

$$F(z)=\sum_{n=0}^{\infty}f(n)z^{-n}$$

$$f(n)=z^{-1}\{F(z)\}$$

关于Z变换有很多具体的计算方法，通常有三种：幂级数展开、部分分式展开和围线积分法。在MATLAB 中，和Z 变换相关的命令采用的是围线积分法，对应的命令是ztrans，其表达式是 $f(n)=\frac{1}{2\pi j}N\oint_{\Gamma}F(z)z^{n-1}dz$。对应的调用命令如下：

- FZ=ztrans(fn,n,z) 求时域函数 fn 的Z变换FZ，fn是以n为自变量的时域序列，FZ是以复频率z为自变量的频域函数；
- fn =iztrans(FZ,z,n) 求频域函数FZ的Z反变换fn，fn是以n为自变量的时域序列，FZ是以复频率z为自变量的频域函数。

5.7.6 Z变换实例

例 5.42 在MATLAB中，对下面的函数实现Z变换。

step 1 在MATLAB的命令窗口输入下面的内容：

```
>> syms n z w a
>> f1 = n^4;
>>f2= a^z;
>>f3=sin(a*n);
>> z1=ztrans(f1);
>> z2=ztrans(f2);
>> z3=ztrans(f3,w);
```

step 2 在MATLAB的命令窗口输入"[simple(z1);simple(z2);simple(z3)]"，查看Z变换后的表达式如下：

```
>> [simple(z1);simple(z2);simple(z3)]
 ans =
  z*(z+1)*(z^2+10*z+1)/(z-1)^5
                     -w/(-w+a)
  w*sin(a)/(w^2-2*w*cos(a)+1)
```

例 5.43 在MATLAB中，对下面的函数实现Z的反变换。

step 1 在MATLAB的命令窗口输入下面的内容：

```
>> syms z n k a
>> f1 = 2*z/(z-2)^2;
>> f2= n*(n+1)/(n^2+2*n+1);
>> f3= z/(z-a);
>> z1=iztrans(f1); z2=iztrans(f2);
>> z3=iztrans(f3);
```

step 2 在MATLAB的命令窗口输入"[z1;z2;z3]"，查看Z反变换后的表达式如下：

```
>> [z1;z2;z3]
 ans =
  2^n*n
 (-1)^k
    a^n
```

符号矩阵的计算

矩阵计算一直都是MATLAB计算的核心内容。关于数值矩阵运算的内容在前面章节中已经介绍过，达里就不重复介绍了。而在MATLAB的符号工具箱中，符号矩阵的运算规则和数值矩阵的运算规则大致相同，没有添加新的运算规则，这就给用户使用符号矩阵带来了极大的方便。本节将主要介绍符号矩阵的一些简单应用。

线性代数运算

符号对象的线性代数运算和双精度的线性代数运算一样，读者可以查阅数值计算的相应内容来了解有关线性代数运算的一些规则和注意事项。在本小节中，将利用符号对象来进行相应的线性代数计算。

例5.44 在MATLAB中使用符号对象进行基础的矩阵运算。

step 1 在MATLAB的命令窗口输入下面的内容：

```
>> syms t
>> [I,J]=meshgrid(1:5);
>> H=1./(I+J-t);
```

在上面的程序代码中，引入了符号对象t，同时产生包含变量t的Hilbert矩阵H，这就相当于在Hilbert矩阵中引入了参数t。

step 2 查看创建的Hilbert矩阵H，如下：

```
>> H
 H =
[   1/(2-t),   1/(3-t),   1/(4-t),   1/(5-t),   1/(6-t)]
[   1/(3-t),   1/(4-t),   1/(5-t),   1/(6-t),   1/(7-t)]
[   1/(4-t),   1/(5-t),   1/(6-t),   1/(7-t),   1/(8-t)]
[   1/(5-t),   1/(6-t),   1/(7-t),   1/(8-t),   1/(9-t)]
[   1/(6-t),   1/(7-t),   1/(8-t),   1/(9-t),  1/(10-t)]
```

step 2 将符号对象t设置为参数1，MATLAB会自动计算出原始的Hilbert矩阵，使用的命令为前面

章节介绍过的Subs，结果如下：

```
>> Hib=Subs(H,t,1)
Hib =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
```



当使用命令subs将符号对象t替换为数值1后，MATLAB会自动进行数值运算，这样原来的符号矩阵就直接转换为数值矩阵Hib。可以将符号t替换为其他的任何数值，MATLAB也会计算出对应的数值矩阵。

step 4 计算符号矩阵H的行列式的倒数：

```
>> d=1/det(H)
 d =
    -1/82944*(-2+t)*(-4+t)^3*(-6+t)^5*(-8+t)^3*(-10+t)*(-9+t)^2*(-7+t)
^4*(-5+t)^4*(-3+t)^2
```

由于H是关于t的符号矩阵，而通过det(H)命令计算该符号矩阵的行列式，该行列式也是关于t的符号表达式，而其对应的倒数d也就是关于t的符号表达式。

step 5 展开上面的符号表达式d，得到的结果如下：

```
>> f=expand(d)
f =
 10640296363350955/96*t^8-323874210240000*t+2885896606895/13824*t^16
-1588946776255/82944*t^17-8194259295156385/
82944*t^13+1115685328012530*t^4+1268467075/864*t^18-240519875/2592*t^19
-15940015/82944*t^21+21896665/4608*t^20+40825/6912*t^22-5375/41472*t^23
-1078920141906600*t^3+742618453752000*t^2+25/13824*t^24-1/82944*t^25
-1970198206236930 25/5184*t^9+3790943429879 3825/3456*t^10
-5560809824710517 5/20736*t^11+672126336000 00-17487546212523 77/
2*t^5+12958201048605475/24*t^6+7707965729450845/13824*t^12 \ -
3882147254934092 5/144*t^7+34372691161375/2304*t^14-79493630114675/
41472*t^15
```

step 6 由于上面的符号表达式d太复杂，所以可以将其简化，结果如下：

```
>> pretty(f)
   10640296363350955   8                             2885896606895   16
  ------------------- t  - 323874210240000 t + -------------- t
         96                                         13824
          1588946776255   17   8194259295156385   13                    4
       - ------------- t   - ----------------- t   + 1115685328012530 t
             82944                  82944
          1268467075   18   240519875   19   15940015   21   21896665   20
       + ---------- t   - --------- t   - -------- t   + -------- t
             864               2592             82944              4608
          40825   22   5375   23                    3                   2
       + ----- t   - ----- t   - 1078920141906600 t  + 742618453752000 t
```

```
     6912          41472
      25    24              25    1970198206236930025  9
   + ----- t    - 1/82944 t    - ------------------- t
     13824                              5184
     37909434298793825  10    55608098247105175  11
  + ----------------- t    - ----------------- t   + 67212633600000
            3456                     20736
                              5    12958201048605475  6   7707965729450845  12
   - 1748754621252377/2 t   + ------------ t   + ----------- t
                                             24                        13824
    38821472549340925  7    34372691161375  14   79493630114675  15
   - ------------ t   + --------- t    - --------- t--------- t
          144                      2304                      41472
```



上面两个步骤中使用的都是符号表达式的相关命令。可以看出，符号矩阵H的相关矩阵运算结果也是符号表达式，符号表达式的命令都可以适用。

step 7 计算H矩阵的逆矩阵X，得到的结果如下：

```
>> X=inv(H)
 X =
 [                          -1/576*(-4+t)^2*(-6+t)^2*(-5+t)^2*(-2+t)*(-3+t)^2,
1/144*(-3+t)*(-6+t)^2*(-7+t)*(-5+t)^2*(-4+t)^2*(-2+t),
-1/96*(-3+t)*(-5+t)^2*(-8+t)*(-7+t)*(-6+t)^2*(-4+t)*(-2+t),
1/144*(-3+t)*(-5+t)*(-7+t)*(-8+t)*(-9+t)*(-6+t)^2*(-4+t)*(-2+t), -1/576*
(-3+t)*(-5+t)*(-7+t)*(-9+t)*(-8+t)*(-6+t)*(-4+t)*(-2+t)*(-10+t)]
[                     1/144*(-3+t)*(-6+t)^2*(-7+t)*(-5+t)^2*(-4+t)^2*(-2+t),
-1/36*(-6+t)^2*(-7+t)^2*(-4+t)*(-5+t)^2*(-3+t)^2,
1/24*(-5+t)*(-8+t)*(-7+t)^2*(-6+t)^2*(-4+t)^2*(-3+t),
-1/36*(-3+t)*(-6+t)*(-9+t)*(-7+t)^2*(-8+t)*(-5+t)^2*(-4+t),
1/144*(-5+t)*(-7+t)*(-9+t)*(-8+t)*(-6+t)^2*(-4+t)*(-3+t)*(-10+t)]
[            -1/96*(-3+t)*(-5+t)^2*(-8+t)*(-7+t)*(-6+t)^2*(-4+t)*(-2+t),
1/24*(-5+t)*(-8+t)*(-7+t)^2*(-6+t)^2*(-4+t)^2*(-3+t),
-1/16*(-4+t)^2*(-8+t)^2*(-6+t)*(-7+t)^2*(-5+t)^2,
1/24*(-4+t)*(-7+t)*(-8+t)^2*(-9+t)*(-5+t)^2*(-6+t)^2,
-1/96*(-4+t)*(-7+t)^2*(-9+t)*(-8+t)*(-6+t)^2*(-5+t)*(-10+t)]
[         1/144*(-3+t)*(-5+t)*(-7+t)*(-8+t)*(-9+t)*(-6+t)^2*(-4+t)*(-2+t),
-1/36*(-3+t)*(-6+t)*(-9+t)*(-7+t)^2*(-8+t)*(-5+t)^2*(-4+t),
1/24*(-4+t)*(-7+t)*(-8+t)^2*(-9+t)*(-5+t)^2*(-6+t)^2,
-1/36*(-6+t)^2*(-8+t)*(-5+t)^2*(-7+t)^2*(-9+t)^2,
1/144*(-5+t)*(-7+t)^2*(-10+t)*(-8+t)^2*(-9+t)*(-6+t)^2]
[ -1/576*(-3+t)*(-5+t)*(-7+t)*(-9+t)*(-8+t)*(-6+t)*(-4+t)*(-2+t)*(-10+t),
1/144*(-5+t)*(-7+t)*(-9+t)*(-8+t)*(-6+t)^2*(-4+t)*(-3+t)*(-10+t),
-1/96*(-4+t)*(-7+t)^2*(-9+t)*(-8+t)*(-6+t)^2*(-5+t)*(-10+t),
1/144*(-5+t)*(-7+t)^2*(-10+t)*(-8+t)^2*(-9+t)*(-6+t)^2,
-1/576*(-8+t)^2*(-7+t)^2*(-6+t)^2*(-10+t)*(-9+t)^2]
```

step 8 计算Hilbert矩阵的逆矩阵，得到的结果如下：

```
>> subs(X,t,1)
ans =
  1.0e+005 *
    0.0003   -0.0030    0.0105   -0.0140    0.0063
   -0.0030    0.0480   -0.1890    0.2688   -0.1260
    0.0105   -0.1890    0.7938   -1.1760    0.5670
```

```
 -0.0140    0.2688   -1.1760    1.7920   -0.8820
  0.0063   -0.1260    0.5670   -0.8820    0.4410
```

在上面的程序代码中，将上面步骤中求得的X矩阵中参数t设置为1，就可以求得Hilbert矩阵的逆矩阵。



在上面的步骤中使用了多种数值矩阵的命令，例如det、inv等。从程序的结果中可以看出，这些常见命令都可以适用于符号表达式中。

5.8.2 特征值运算

特征值运算一直是矩阵运算的重要内容。在符号矩阵中，同样提供数值矩阵类似的求解特征值的命令eig。同时，和它们对应的任意精度计算的命令则是eig(Vpa(A))。

例 5.45 在MATLAB中，使用符号对象来进行矩阵运算。

step 1 在MATLAB的命令窗口输入下面的内容：

```
>> A=sym(gallery(5));
>> B=A^5;
```

在上面的程序代码中使用代码sym(gallery(5))创建一个5阶的符号矩阵A，然后对该符号矩阵A进行乘幂计算得到矩阵B。

step 2 查看乘幂的结果矩阵B：

```
>> B
 B =
[ 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0]
```

可以看出，A的5阶乘幂是零矩阵。根据线性代数的基础知识，A是零幂矩阵，其对应的所有特征值都应该是0。

step 3 求解符号矩阵A的特征多项式：

```
>> p=poly(A,'lambda')
 p =
 lambda^5
```

在上面的程序代码中，使用poly命令来求解符号矩阵A对应的特征多项式，由于矩阵A的特征值为0，因此其特征多项式就是一个5阶的表达式。

step 4 求解符号矩阵A的特征值：

```
>> lambda=eig(A)
 lambda =
 0
 0
 0
```

```
0
0
```

从上面的程序结果中可以看出，通过符号矩阵的eig命令，可以直接求解数值矩阵的特征值，得到的结果为 0。

step 5 求解任意精度下符号矩阵 A 的特征值，得到的结果如下：

```
>> lambda=eig(vpa(A))
 lambda =
  -.22428967199963540318045946788 22e-5+.16295641216458292389044088574558e-5*i
 -.2242896719996354031804594678822e-5-.16295641216458292389044088574558e-5*i
   .8567143734873483165916892462 87e-6+.26366776411426003145716139438407e-5*i
   .856714373487348316591689246287e-6-.26366776411426003145716139438407e-5*i
                                              .2772364693018011430425880323080e-5
```

step 6 求解符号矩阵 A 的约当(Jordan)标准型，得到的结果如下：

```
>> J=jordan(A)
 J =
[  0,  1,  0,  0,  0]
[  0,  0,  1,  0,  0]
[  0,  0,  0,  1,  0]
[  0,  0,  0,  0,  1]
[  0,  0,  0,  0,  0]
```

step 5 求解符号矩阵 A 的矩阵指数，得到的结果如下：

```
>> E=expm(t*A)
 E =
 [                          1-9*t+11/2*t^2-2/3*t^3,
4/3*t^3-9*t^2+11*t,                              -10/3*t^3+39/2*t^2-21*t,
32/3*t^3-58*t^2+63*t,                            -85/2*t^3+232*t^2-252*t]
[                  70*t-115*t^2+81/2*t^3-7/2*t^4,
1+7*t^4-67*t^3+301/2*t^2-69*t,           -35/2*t^4+293/2*t^3-299*t^2+141*t,
56*t^4-438*t^3+1799/2*t^2-421*t,         -1785/8*t^4+3503/2*t^3-3597*t^2+1684*t]
[             -575*t+1717/2*t^2-285*t^3+71/3*t^4,
-142/3*t^4+1426/3*t^3-1146*t^2+575*t,        1+355/3*t^4-3139/3*t^3+4585/2*t^2-1149*t,
-1136/3*t^4+3140*t^3-6875*t^2+3451*t,        6035/4*t^4-75323/6*t^3+27496*t^2-13801*t]
[           3891*t-5837*t^2+11675/6*t^3-973/6*t^4,
973/3*t^4-3243*t^3+15565/2*t^2-3891*t,      -4865/6*t^4+14269/2*t^3-15565*t^2+7782*t,
1+7784/3*t^4-64210/3*t^3+93391/2*t^2-23345*t,
-82705/8*t^4+513437/6*t^3-373503/2*t^2+93365*t]
[                1024*t-1536*t^2+512*t^3-128/3*t^4,
256/3*t^4-2560/3*t^3+2048*t^2-1024*t, -640/3*t^4+5632/3*t^3-4096*t^2+2048*t,
2048/3*t^4-5632*t^3+12288*t^2-6144*t,      1-2720*t^4+67552/3*t^3-49144*t^2+24572*t]
```

step 8 求解符号矩阵 A 的矩阵元素指数，得到的结果如下：

```
>> X=exp(t*A)
 X =
[     exp(-9*t),      exp(11*t),      exp(-21*t),      exp(63*t),    exp(-252*t)]
[     exp(70*t),      exp(-69*t),     exp(141*t),      exp(-421*t),   exp(1684*t)]
[   exp(-575*t),     exp(575*t),   exp(-1149*t),     exp(3451*t), exp(-13801*t)]
[   exp(3891*t),  exp(-3891*t),     exp(7782*t), exp(-23345*t),   exp(93365*t)]
```

```
[   exp(1024*t),  exp(-1024*t),   exp(2048*t),  exp(-6144*t),  exp(24572*t)]
```



在上面的程序中分别使用expm和exp命令来求解矩阵指数和元素指数数值。对于两个命令的差别，请查看相应的帮助文件。

5.9 符号代数方程的求解

在代数和数学分析内容中，求解方程一直都是十分重要的内容。从代数理论的角度来讲，一般代数方程包括线性、非线性和超越方程等。本节将主要介绍符号代数方程的命令和实例。

5.9.1 solve 命令

对于上述的方程类型，MATLAB提供统一的求解命令solve。当方程组不存在符号解，同时如果没有其他的自由参数时，solve命令会给出数值解。

在MATLAB中，solve命令的调用格式如下。

- g = solve(eq)　其中eq可以是符号表达式或不带符号的字符串表达式，该函数的功能是求解eq=0的方程，其中自变量采用默认变量，通过函数findsym函数来确定。
- g = solve(eq,var)　求解方程eq=0，其自变量由参数var指定。其中eq和上一种调用方式相同。返回值g是由方程的所有解所构成的列向量。
- g = solve(eq1,eq2,...,eqn)　求解符号表达式或不带符号的字符串表达式eq1，eq2，...，eqn所构成的方程组，其中自变量采用默认变量，可以通过函数findsym函数来确定。
- g = solve(eq1,eq2,...,eqn,var1,var2,...,varn)　求解符号表达式或不带符号的字符串表达式eq1，eq2，...，eqn所构成的方程组，其中自变量由参数var1，...，varn确定。



当输出参数的数目和方程组的数目相等的时候，方程组的解将分别赋给每个输出参数，并且按照字母表的顺序进行排列。

5.9.2 求解非线性方程组

例 5.46　求解非线性方程组$\begin{cases} x^2 - xy + y = 3 \\ x^2 - 4x + 3 = 0 \end{cases}$的数值解。

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> syms x y
>> [x,y] = solve('x^2 + x*y + y = 3','x^2 - 4*x + 3 = 0');
```

step 2 在命令窗口中输入结果矩阵，得到的结果如下：

```
>> solution=[x,y]
    solution =
[     1,     1]
[     3, -3/2]
```



从上面的求解结果可以看出，MATLAB会成对地输出方程组的数值解。如果没有引入新的变量，则会分别输出x和y的结果。因此，建议引入新的变量。

5.9.3 求解含参数方程组

例 5.47 求解含有参数 a 的非线性方程组 $\begin{cases} au^2+v^2=0 \\ u-v=1 \end{cases}$ 的解。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> syms u v a
>> [u,v]=solve('a*u^2+v^2=0','u-v=1');
>> solution=simple([u,v]);
```

step 2 在命令窗口中输入结果矩阵，得到的结果如下：

```
>> solution
 solution =
[   ((-a)^(1/2)+1)/(a+1), -(a-(-a)^(1/2))/(a+1)]
[  -((-a)^(1/2)-1)/(a+1), -(a+(-a)^(1/2))/(a+1)]
```

5.9.4 求解超越方程组

例 5.48 求解超越方程组 $\begin{cases} \sin(x+y)-e^x y=0 \\ x^2-y=2 \end{cases}$ 的解。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> syms x y
>> S=solve('sin(x+y)-exp(x)*y=0','x^2-y=2');
```

step 2 在命令窗口中输入结果矩阵，得到的结果如下：

```
>> S
S =

    x: [ 2x1 sym]
    y: [ 2x1 sym]
```

上面的程序结果中，并没有显示方程组的解，只是显示方程结果的属性和维度。在本例中，变量 x 和 y 都是符号变量，维度都是 2x1。

step 3 查看各个变量的具体数值，得到的结果如下：

```
>> S.x
 ans =
  -.33129879499763797066864098166363
 -.66870120500236202933135901833637
>> S.y
 ans =
  -1.8902411084331130499424622177919
 -1.5528386984283889912797441811191
```



如果没有指定该方程组的变量，MATLAB会使用findsym函数依次将x、y作为方程的自变量，最终的结果会和本实例的结果有所差别，请读者自行尝试。

5.10 符号微分方程的求解

相对于符号线性代数方程组的求解，微分方程的求解稍微复杂一点。限于本章的篇幅，只介绍如何使用MATLAB求解常微分方程组。MATLAB同样可以求解偏微分方程组，但是求解起来十分复杂。

前面曾经介绍微分方程的数值求解方法，与初值问题求解相比，微分方程边值问题的求解显得复杂和困难。但是，对于符号求解而言，不论初值问题还是边值问题，其求解微分方程的指令都很简单。另一方面，符号微分方程的计算可能会花费较多的时间，可能得不到简单的解析解，可能得不到封闭形式的解，甚至可能得不到解。从这个角度来看，微分方程的数值解和解析解是互补的方法。

5.10.1 dsolve 命令

本小节将介绍如何使用符号微分方程进行求解。在MATLAB中，dsolve命令的常见调用格式如下：

- r = dsolve('eq1,eq2,...', 'cond1,cond2,...', 'v')
- r = dsolve('eq1','eq2',...,'cond1','cond2',...,'v')

上面两个命令的功能是：求由eq1、eq2……指定的常微分方程的符号解。常微分方程以变量v作为自变量，参数cond1、cond2……用来指定方程的边界条件或者初始条件。如果不指定v作为自变量，则将默认的t作为自变量。



和solve命令类似，dsolve命令也有更为简单的形式。但是根据笔者的经验，建议在理解微分方程组的独立变量以及次序理解充分的时候，才使用简单的形式。

5.10.2 求解常微分方程

例 5.49 求常微分方程 $\frac{dx}{dt}=-ax$ 的通解，然后求解在初值条件 $x(0)=1$ 的数值解。

step 1 在MATLAB的命令窗口中输入下面的命令：

```
>> syms x a
>> S1=dsolve('Dx=-a*x');
>>S2=dsolve('Dx=-a*x','x(0)=1');
>> solution=[S1;S2];
```

step 2 在命令窗口中输入"solution"，查看求解的结果：

```
>> solution
 solution =
  C1*exp(-a*t)
    exp(-a*t);
```

在上面的程序代码中，S1 是微分方程的通解，S2 则是初值条件下的特解。在默认情况下，MATLAB 会依次使用 C1、C2 作为通解的常量符号。

5.10.3 求解二阶常微分方程

例 5.50 求二阶常微分方程 $\begin{cases}\dfrac{d^2y}{dx^2}=\cos(2x)-y\\ y(0)=1,y'(0)=0\end{cases}$ 的通解。

step 1 在 MATLAB 中的命令窗口中输入下面的命令：

```
>> syms x y
S=dsolve('D2y=cos(2*x)-y','y(0)=1','Dy(0)=0','x');
```

step 2 在命令窗口中输入"S"，查看求解的结果：

```
>> S
 S =
4/3*cos(x)-1/3*cos(2*x)
```

在上面的程序中，求解的方程为 $y(x)$，而不是方程 $y(t)$，如果在命令中没有特别指明方程自变量 x，得到的结果将是关于自变量 t 的表达式，如下：

```
>> S1=dsolve('D2y=cos(2*x)-y','y(0)=1','Dy(0)=0');
>> S1
 S1 =
 cos(t)*(1-cos(2*x))+cos(2*x)
```



在常微分方程中，用大写字母 D 表示一次微分，则 D2、D3 分别表示二次、三次微分运算。再次提醒读者，MATLAB 是区分大小写的，只能用 D 表示微分，使用小写字母 d 就无法辨识该符号。

5.10.4 求解常微分方程组

例 5.51 求常微分方程组 $\begin{cases}\dfrac{df}{dt}=3f(t)+4g(t)\\ \dfrac{dg}{dt}=-4f(t)+3g(t)\\ f(0)=0,g(0)=1\end{cases}$ 的通解。

step 1 在 MATLAB 中的命令窗口中输入下面的命令：

```
>> syms f g
>> [f,g]=dsolve('Df=3*f+4*g,Dg=-4*f+3*g','f(0)=0,g(0)=1');
```

step 2 查看上面微分方程组的求解结果：

```
>> disp('f=');disp(f)
f=
exp(3*t)*sin(4*t)
```

```
>> disp('g=');disp(g)
g=
exp(3*t)*cos(4*t)
```



如果不使用符号微分方程组，而使用数值方程的方法来求解，相应的求解方法相比较而言则会非常复杂。

5.11 利用 Maple 的资源

在 Maple 软件中，有 2000 多条符号计算命令。前面已经多次使用 Maple 的相关命令，但是这些都只涉及最常见的命令。在 MATLAB 中，为了进一步利用 Maple 软件的符号计算能力，可以利用多种 MATLAB 和 MAPLE 中的接口程序。其中，sym 和 maple 可以直接调用 maple 中计算程序，而 mhelp、mfun 等命令则可以对 maple 的相关函数进行查询和查看等。下面分小节详细介绍。

5.11.1 调用 maple 的相关命令

在 MATLAB 中，用户可以使用 sym 命令定义 maple 软件中的相关命令，然后使用设置后的符号变量，调用相应的相关命令。

例 5.52 在 MATLAB 中求解数值的阶乘。

step 1 在 MATLAB 中的命令窗口中输入下面的命令：

```
>> kfrac=sym('k!');
>> syms k n
>> nfrac=subs(kfrac,k,n);
```

step 2 查看程序完成的结果：

```
>> nfrac
 nfrac =
 n!
```

可以看出，当需要在 MATLAB 中引用 maple 中的命令时，不能直接调用该命令，而只能用 MATLAB 中相关的符号命令来变相引用该命令。

step 3 利用上面的程序计算的数值，得到结果：

```
>> fivefrac=subs(kfrac,k,5)
    fivefrac =
 120
```

例 5.53 在 MATLAB 中利用 maple 中的 "rsolve" 命令，求解递归方程的通解。

step 1 利用 maple 中的命令求解递归方程 $\begin{cases} y(n)y(n-1)+y(n)-y(n-1)=0 \\ y(0)=a \end{cases}$ 的通解，在 MATLAB 的命令窗口中输入下面的内容：

```
>> R=maple(' rsolve({ y(n)*y(n-1) + y(n) - y(n-1) = 0, y(0)=a}, y)');
>> clR=class(R);
```

在上面的程序中，使用maple命令直接调用rsolve命令，求解上面的递归方程，然后使用class命令查看R变量的类别。

step 2 在命令窗口中输入下面的内容，查看上面命令得到的结果：

```
>> R
R =
a/(1+n*a)
>> clR
clR =
char
```



从上面的结果可以看出，通过maple命令MATLAB可以求解递归方程的结果，而且该结果的变量属性是“字符串”，而不是符号对象。

step 3 将上面的结果转换为符号对象，并修改其显示效果：

```
>> M=sym(R);
>> pretty(M)
                                        a
                                     -------
                                     1 + n a
```

step 4 利用maple中的命令求解递归方程$\begin{cases} F(n)=F(n-1)+F(n-2) \\ F(1)=1, F(2)=1 \end{cases}$的通解，在MATLAB的命令窗口中输入下面的内容：

```
>> S=maple('rsolve({F(n) = F(n-1) + F(n-2), F(1..2)=1},F)');
>> Sr=sym(S);
>> pretty(Sr)
                 1/2                1/2 n          1/2                1/2 n
        - 1/5 5     (1/2 - 1/2 5     )    + 1/5 5     (1/2 + 1/2 5     )
```



在上面的程序中，字段“F(1..2)=1”代表的就是初值条件F(1)=1、F(2)=1，这种表达方法是在软件maple中的用法。

step 5 查看该表达式的初值条件，在MATLAB的命令窗口中输入下面的内容：

```
>> syms n
>> F1=subs(Sr,n,1);
>> F2=subs(Sr,n,2);
>> [F1;F2]
ans =
    1
    1
```

在上面的程序中，将符号表达式Sr中的n替换为1和2，就相当于求解表达式的初值条件F(1)和F(2)，得到的结果都是1，可见表达式符合初值条件。

查看maple的帮助

上面已经介绍如何在MATLAB中调用maple中的命令，限于篇幅，不能介绍maple中的所有命令。如果需要了解某个命令在maple的使用方法，则需要使用到maple的帮助。本小节中，将介绍如何在MATLAB中获得maple的帮助。

例 5.54 在MATLAB环境中查阅maple的各种帮助文件。

step 1 查看关于命令"mhelp"的帮助，输入"mhelp"，然后按"Enter"键，得到的结果如下：

```
>> mhelp
 MHELP  Maple help.
    MHELP topic prints Maple's help text for the topic.
    MHELP('topic') does the same thing.
    MHELP is not available with MATLAB Student Version.
    Example:
       mhelp gcd
    Reference page in Help browser
       doc mhelp
```

step 2 查阅maple在线帮助的索引条目，输入"mhelp index"命令，然后按"Enter"键，得到的结果如下：

```
>> mhelp index
 Index of help descriptions
Calling Sequence
     ?index[ category]    or    help(index, category)
Description
- The following categories of topics are available in the help subsystem:
    expression   operators for forming expressions
    function     list of Maple functions
    misc         miscellaneous facilities
    module       topics related to modules
    packages     descriptions of library packages
    procedure    topics related to procedures and programming
    statement    list of Maple statements
  To access these help pages, you must prefix the category with index,
thus ?
  index[ category] .
```

step 1 查阅maple中的具体分类目录，输入"mhelp index[function]"命令，然后按"Enter"键，得到的结果如下：

```
>> mhelp index[ function]
 Index of descriptions for standard library functions
Description
- The following are the names of Maple's standard library functions.
For more
  information, see ?f where f is any of these functions.
    about               abs              add              addcoords
    additionally        addproperty      addressof        AFactor
    AFactors            AIrreduc         AiryAi           AiryAiZeros
    AiryBi              AiryBiZeros      algebraic        algsubs
    bernstein           BesselI          BesselJ          BesselJZeros
```

```
Svd                   symmdiff             symmetric         syntax
system                table                tan               tanh
verify                WeberE               WeierstrassP      WeierstrassPPrime
WeierstrassSigma      WeierstrassZeta      whattype          WhittakerM
WhittakerW            with                 worksheet         WRAPPER
writebytes            writedata            writeline         writestat
writeto               zero                 Zeta              zip
ztrans                _
See Also
libname, index[ package]
```

step 4 查看具体函数的帮助文件，在MATLAB中输入"mhelp rsolve"，然后按"Enter"键，查看查询结果：

```
>> mhelp rsolve
 rsolve - recurrence equation solver
Calling Sequence
    rsolve(eqns, fcns)
    rsolve(eqns, fcns, 'genfunc'(z))
    rsolve(eqns, fcns, 'makeproc')
Parameters
    eqns - single equation or a set of equations
    fcns - function name or set of function names
    z    - name, the generating function variable
Description
- The function rsolve attempts to solve the recurrence relation(s)
specified in eqns for the functions in fcns, returning an expression
for the general term of the function.
.....................................................................
Examples
> rsolve(f(n) = -3*f(n-1) - 2*f(n-2), f(k));
                                     k                       k
                 (2 f(0) + f(1)) (-1)  + (-f(0) - f(1)) (-2)
.....................................................................
See Also
asympt, dsolve, genfunc, msolve, solve
```

从上面的查看结果中可以看出，关于一般函数的帮助信息包括函数的表达式、参数说明、描述和函数实例等，可以查阅其他函数的帮助信息。

可视化符号分析

在MATLAB中，数学工具箱为符号函数的可视化提供一组简易的命令，本节将着重介绍两个数学分析的可视化界面：单变量函数分析界面和泰勒级数逼近分析界面。其中，单变量函数分析界面由命令funtool引出，泰勒级数逼近分析界面由命令taylortool引出；引出界面之后，后续的所有操作都可以直接在界面上进行，下面分别详细介绍。

单变量函数分析界面

在MATLAB中，单变量函数分析界面主要用来分析单变量函数的关系，最多可以分析两个函数之

间的关系。对于习惯使用计算器或者只做一些简单的符号计算和图形处理的读者，该分析界面是一个很好的选择。该计算器功能简单，操作十分方便，可视性强。其调用格式为：funtool。当用户输入该命令后，MATLAB 会调用如图 5.4 所示的界面。

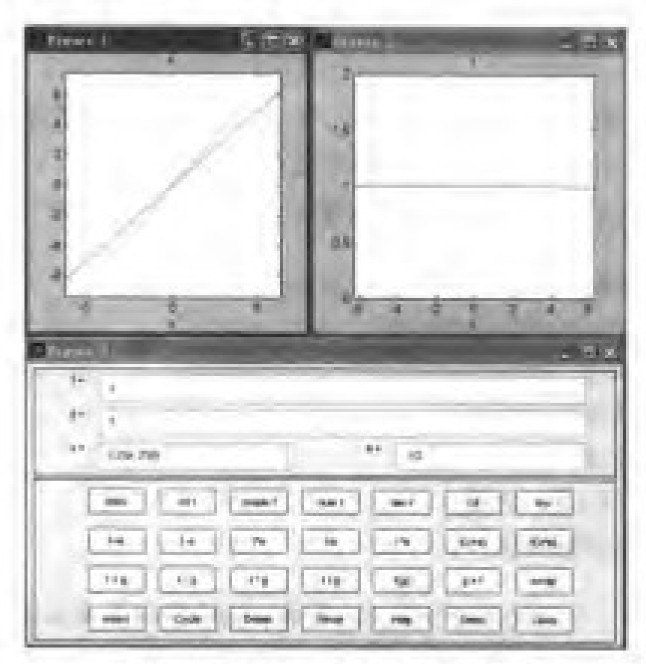


图 5.4 单变量函数分析的默认界面

上面的界面是 MATLAB 调用单变量函数分析的默认界面。可以看出，在默认情况下，函数 f=x，g=1，同时自变量 x 的取值范围是 [-2*pi, 2*pi]，常数 a=1/2。同时可以看出，这个函数界面由两个图形窗口和一个函数运算控制窗口共三个独立窗口组成。在任何时候，两个图形窗口只有一个处于激活状态，而函数运算控制窗口上的操作只能对被激活的函数图形窗口起作用。

为了演示如何使用该运算界面，在本小节中将修改 f 和 g 的函数表达式，查看各种操作的结果和图形的变化，修改后的界面如图 5.5 所示。

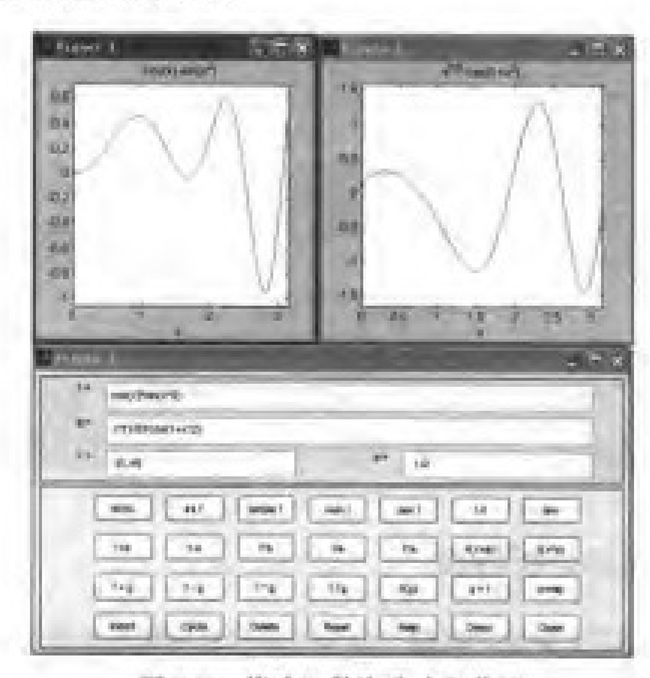


图 5.5 修改函数表达式和范围

在函数操作界面中，第一排的按钮都是针对函数 f 的，功能依次为求导（对应 "df/dx" 按钮）、积分（"inf" 按钮）、简化（"simple" 按钮）、提取函数的分子（"numf" 按钮）、提取函数表达式的分母（"denf" 按钮）、求倒数（"1/f" 按钮）和求反函数（"invf" 按钮）等。

第二排的按钮主要用来处理函数 f 和常数 a 之间的运算，包括加、减、乘、除等，对应的按钮中的符号表示对应的操作。

第三排的按钮主要用来处理两个函数 f 和 g 之间的运算。其他按钮的功能可以直接从按钮的符号了解到，最后一个按钮 "Swap" 的功能是实现两个函数的转换。

最后一排的按钮主要用来进行计算器自身的操作，下面详细介绍各个按钮的具体功能。

- Insert：将当前激活窗口中的函数插入函数列表 fxlist 中。
- Cycle：依次循环显示 fxlist 中的函数。
- Delete：从 fxlist 列表中删除当前激活窗口中的函数。
- Reset：将计算器恢复到初始状况。
- Help：关于该界面的所有帮助内容。
- Demo：关于该界面的演示内容。
- Close：关闭该界面的所有窗口。

最后，该界面实际上是一个已经制作好的GUI界面，如果希望查看其源程序，或者在其基础上修改源程序，可以首先查看其源程序。单击界面中的 "Help" 按钮，MATLAB 会调用相关的命令，然后单击 "View code for funtool" 选项，打开该 GUI 界面对应的 M 文件，如图 5.6 所示。

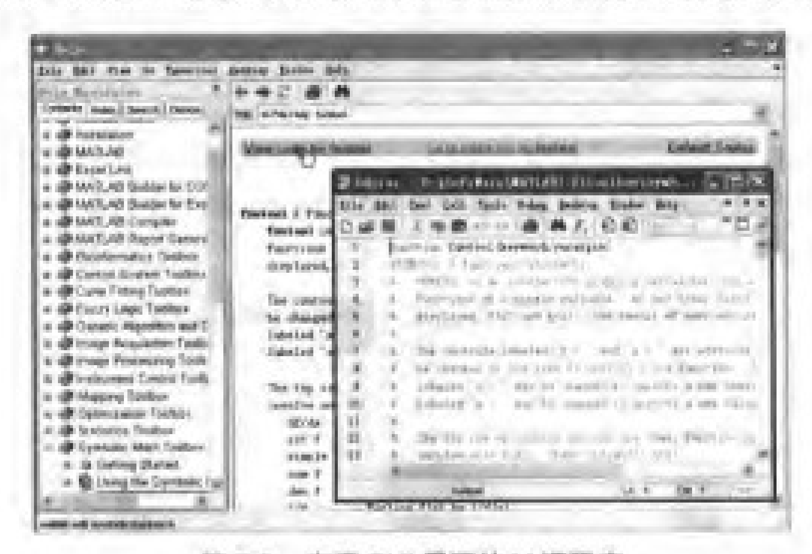


图 5.6　查看 GUI 界面的 M 源程序



根据前面内容的介绍，读者也许已经发现，该数学分析界面中的按钮功能针对函数 f 的居多，针对函数 g 的居少，但是可以使用 "Swap" 按钮转换两个函数的位置。

5.12.2 泰勒级数逼近分析界面

在 MATLAB 的符号工具箱中，为用户提供 "taylor" 命令来求解符号表达式的泰勒级数展开式，使用该命令可以求解任何符号表达式在任意数据点的泰勒级数展开式。

泰勒级数逼近是一种十分常见的函数分析方法，该工具的功能主要是分析某范围内的函数形态，

因此，如果能够绘制对应范围内的函数图形，可以更加直观地分析函数的泰勒级数逼近和原来函数的性态。

为了满足上面的条件，MATLAB提供泰勒级数逼近分析界面，调用该界面的命令为taylortool。当用户在命令窗口中输入该命令后，MATLAB会自动调用界面，如图5.7所示。

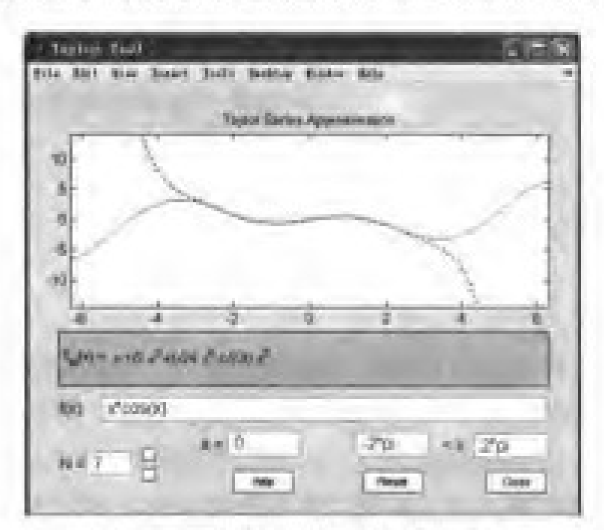


图 5.7 泰勒级数分析的默认界面

在上面的界面中，a表示级数的展开点，N表示级数展开阶数，可以通过右侧的按钮来改变数值，也可以在选框中输入阶的数值。函数级数展开的默认范围是(−2π,2π)，可以在对应的选框中修改范围的数值。

可以修改级数展开的所有参数，查看界面的对应变化，如图5.8所示。

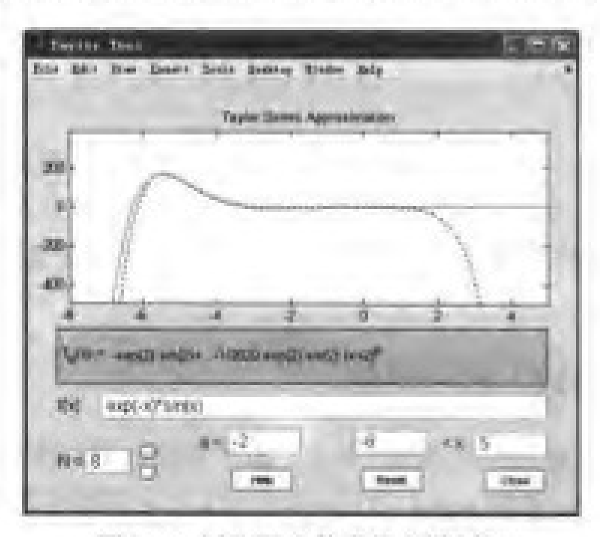


图 5.8 对不同函数进行泰勒展开

5.13 小结

本章向读者介绍了关于符号计算的详细内容，包括符号对象、表达式、符号操作、符号代数方程、符号微分方程等。了解了这些内容后，基本上可以对MATLAB中如何实现符号运算有一个大概的了解。在后面的章节中，将介绍如何在MATLAB中实现数据和函数的可视化。

◆ ◆ ◆

第6章 数据和函数的可视化

本章包括

- 绘制二维曲线
- 编辑二维曲线的属性
- 绘制三维曲线图
- 绘制三维曲面图
- 绘制特殊图形
- 绘制四维图形
- 编辑三维图形的属性
- 绘制复数图形
- 图形的打印和输出

在前面的章节中，已经介绍和分析了MATLAB在数据处理、运算和分析中的各种运用。和其他科学计算工具类似，MATLAB也提供强大的图形编辑功能。通过图形，用户可以直观地观察数据间的内在关系，也可以十分方便地分析各种数据结果。从最初的版本开始，MATLAB就一直注重数据的图形表示，而且在更新版本的时候不断地使用新技术来改进和完善可视化的功能。

在本章中，将详细介绍MATLAB中的图形形成原理，曲线、曲面绘制的基本技巧，如何编辑图形参数，如何标记图形等。这些操作大部分只涉及MATLAB中的高层命令，这些命令格式简单，容易理解。对于底层的图形操作，将在后面章节中详细介绍。

最后，相对于MATLAB的6.x版本，MATLAB7提供功能十分强大、使用非常方便的图形编辑功能，使得原来十分复杂的图形编辑功能变得十分简单。这些内容将在本章的最后部分加以介绍。

图形的基础知识

在正式学习和熟悉MATLAB的图形编辑功能之前，有必要了解MATLAB中图形绘制的基本原理和基础步骤。本节大致将MATLAB图形的数据源分成离散和连续两个部分，介绍在MATLAB中如何绘制这两种图形。由于是介绍基础知识，所以选取的函数形式比较简单，主要是为了介绍绘制图形的原理和步骤。

离散数据（函数）的可视化

在MATLAB中，绘制离散数据的原理十分简单。对于离散数据数组$x=[x_1, x_2, x_3,..., x_n]$，$y=[y_1, y_2, y_3,..., y_n]$，MATLAB会将对应的向最组$(x_1,y_1)$，$(x_2,y_2)$，… ，$(x_n,y_n)$用直角坐标中的点序列表示。一般情况下，MATLAB中的离散序列所反映的只是某确定区间的对应关系（也就是函数关系），不能反映无限区间上的对应关系。

例6.1 绘制离散函数$y=\frac{1}{(n-3)^2+1}+\frac{1}{(n-9)^2+4}$的图形，其中自变量$n$是取值范围是（0,16）的整数。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> n=0:16;
>> y = 1 ./ ((n-3).^2 + 1) + 1 ./ ((n-9).^2 + 4) + 5;
>> plot(n,y,'mh','markersize',15);
```

```
>> axis([0,17,5,6.2]);
>> grid on
```

step 2 按"Enter"键，MATLAB会弹出名为"Figure 1"的图形对话框，如图6.1所示。

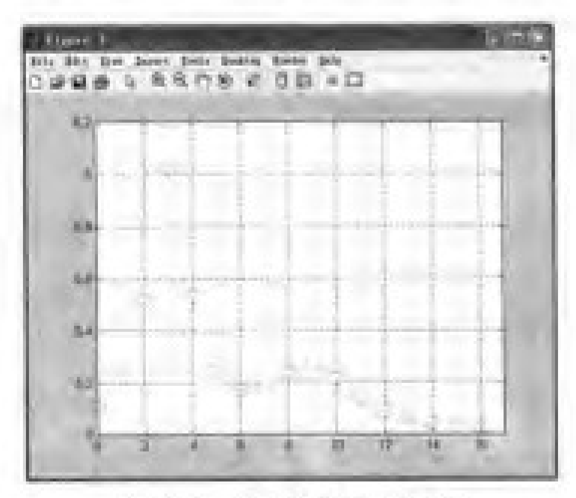


图6.1 离散函数的可视化

上面的程序语句十分简单，但是已经涉及MATLAB中图表绘制的多个方面。下面详细介绍各个语句的含义。

- 程序语句"plot(n,y,'mh','markersize',15);"是图表绘制的主要命令。其中，第三个参数'mh'的作用是设置数据点的色彩和点形，m表示颜色是"品红"，h表示数据点形是"六角星符"，关于其他颜色和点形参数将在后面章节中详细介绍；上面语句中的第四个参数'markersize'用来设置标记的大小属性，15就是其属性值。
- 程序语句"axis([0,17,5,6.2])；"的功能是设置图表坐标轴的范围。其中，横坐标轴（X轴）的刻度范围是[0,17]；纵坐标轴（Y轴）的刻度范围是[5,6.2]。在MATLAB中对坐标轴的其他设置将在后面章节中详细介绍。
- 程序语句"grid on"的功能是添加分格线。在默认情况下，MATLAB会为所有坐标轴都添加分格线；如果需要单独对其中某个坐标轴添加分格线，需要使用句柄图形的内容，将在后面的章节中介绍。



在默认情况下，当用户在命令窗口中输入plot命令后，MATLAB程序就会将图表显示在"Figure 1"的对话框中，而且该对话框会和MATLAB的操作界面独立。如果希望将图表在命令窗口中显示，可以单击菜单栏中的按钮。

6.1.2 连续函数的可视化

在MATLAB中，绘制连续函数的图表原理基本和离散函数相同，也就是在一组离散自变量上计算相应的函数值，然后将各个数据组用点来表示。这似乎和离散函数相比，没有太大的区别。在MATLAB中，将这些离散数据点转化为连续函数的方法是：减少离散数据点的间隔，增加更多的数据点，计算各个数据点的函数数值，这个方法相当于微分的思想；另外一个方法就是直接将相邻的数据点用直线连接起来，用线性关系来替代其他的函数关系。



在MATLAB中，使用上面两个方法都可以绘制连续函数的图形。如果函数的自变量采样点数不够多，两种方法绘制的连续图形都会有很大的误差。

例 6.2 绘制函数$y=\frac{1}{(x-3)^2+1}+\frac{1}{(x-9)^2+4}+5$的图形，其中自变量的取值范围是［0,16］的所有实数。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> x1=0:16;
>>y1 = 1 ./ ((x1-3).^2 + 1) + 1 ./ ((x1-9).^2 + 4) + 5;
>> x2=0:0.02:16;
>>y2 = 1 ./ ((x2-3).^2 + 1) + 1 ./ ((x2-9).^2 + 4) + 5;
>> subplot(2,2,1),plot(x1,y1,'rp'),axis([0,17,5,6.2]),title('Picture 1'),
grid on
>> subplot(2,2,2),plot(x2,y2,'rp'),axis([0,17,5,6.2]),title('Picture 2'),
grid on
>> subplot(2,2,3),plot(x1,y1,x1,y1,'rp'),axis([0,17,5,6.2]),title
('Picture 3'),grid on
>> subplot(2,2,4),plot(x2,y2,'LineWidth',2),axis([0,17,5,6.2]),title
('Picture 4'),grid on
```

step 2 按"Enter"键，就可以得到相应的结果，如图 6.2 所示。

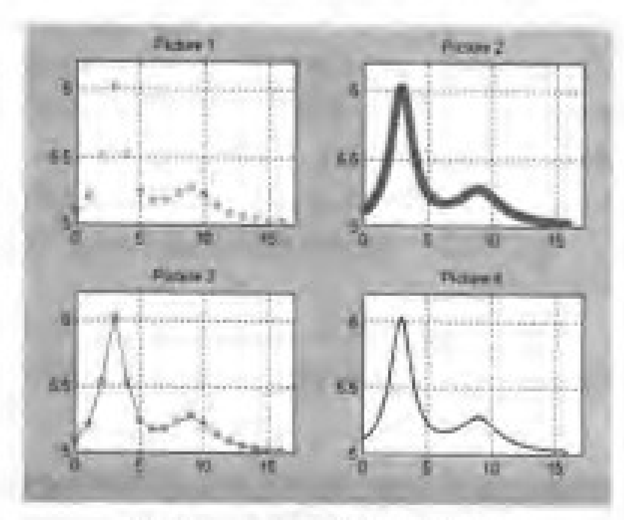


图 6.2 连续函数的表示方法

step 3 分析图形结果。

上面的图表命令中，Picture1 和Picture3使用的是数组x1 和y1，Picture2和Picture4 使用的是图表数组 x2 和 y2。从上面的对比可以很明显的看出，当自变量使用的数组离散数据间隔变小后，图表可以更明显地表示出函数的形态。

Picture 1 和Picture 3尽管也表现出了大概的形态，但是在图表的局部有明显的失真，不能很好表示函数性态。

从上面的程序中可以看出，对于离散数据，MATLAB的plot命令在默认情况下会自动将这些离散数据用直线连接起来，使之成为连续曲线。



在上面的程序语句中，为了对比效果更加明显，将四幅图形绘制在同一个图形窗中，因此，所有的图形就变成了子图形。在MATLAB中，绘制子图表的命令是subplot，该命令在实际应用中会经常用到，会在后面的章节中详细介绍其用法。

6.1.3 绘制图表的基础步骤

从前面两小节的内容中，已经基本了解了在 MATLAB 中绘制图表的方法。在本小节中，将归纳 MATLAB 中绘制图表的基本步骤如下。

step 1 准备图表的数据。这是基础的步骤，因为图表就是用来显示绘图数据变化规律的，因此，首先需要为绘制图表准备数据。一般而言，用户需要首先确定图表绘制的范围，然后选择对应范围的自变量，最后计算对应的函数数值。

step 2 设置显示图表的位置。这个步骤主要针对多子图的情况。当用户绘制多子图的时候，则需要设置每个子图形的显示位置。对应的命令就是上面小节中使用过的 subplot 命令，可以使用该命令指定相应的窗位。

step 3 绘图，并设置相应的参数。这是在 MATLAB 中绘图的主要步骤，对应的命令就是 plot，用户可以使用该命令绘制图表，同时可以设置图表的参数，例如线型、颜色和数据点形等。完成该命令后，基本上就完成了图表的大致外观。

step 4 设置坐标轴属性。从这个步骤开始，开始对图表的编辑工作。用户可以设置坐标轴的刻度范围，添加坐标轴的分格线等。

step 5 添加图形注释。完成图表的基础外观并设置坐标轴属性后，还可以添加一些注释信息，例如图表的标题、坐标轴名称、图例和文字说明等。对于这些注释，MATLAB 都提供对应的命令。用户可以使用命令，方便地设置各种注释。



在上面的步骤中，步骤 1 和步骤 3 是最基础的，通过这两个步骤基本上就可以完成图表的绘制工作，其他步骤都可以认为是对图表的编辑步骤。

6.2 绘制二维曲线

在 MATLAB 中，二维曲线和三维曲线在绘制方法上有较大的差别；相对而言，绘制二维曲线比三维曲线要简单。因此，本节介绍如何在 MATLAB 中绘制二维曲线。

6.2.1 plot 命令

在 MATLAB 中，绘制二维曲线的最基本的命令是 plot，其他的绘制命令都是以 plot 为基础的，而且调用方式都和该命令类似，因此，在本小节中将详细介绍 plot 的使用方法。

在 MATLAB 中，调用 plot 的方法有下面 3 种。

- plot(X,'PropertyName',PropertyValue,...)

  其中参数 X 表示的是绘制图表的数据，'PropertyName' 表示图表属性的选项字符串，例如，线

型、颜色和数据点形等；PropertyValue表示对应属性的选值。对于参数X的不同类型，MATLAB会作不同的处理，详细内容如下：

- 当X是实数向量时，MATLAB会以X向量元素的下标为横坐标，元素数值为纵坐标绘制连续曲线；
- 当X是实数矩阵时，MATLAB会绘制矩阵中每列数值元素相对于其下标的连续曲线，因此绘制的图表结果中包含多个图表，个数等于X矩阵的列数；
- 当X是复数矩阵的时候，以列为单位，分别以矩阵元素的实部为横坐标，以元素的虚部为纵坐标绘制连续曲线。

◆ plot(X,Y,'PropertyName',PropertyValue,...)

和上面的命令相比，该命令多了一个参数Y，其中X，Y都是图表的数据数组。其他参数的含义和上面的命令相同。同样，对于参数X、Y的不同数据类型，MATLAB也会作不同的处理，详细内容如下：

- 当X、Y是同维向量时，MATLAB会以X和Y元素为横、纵坐标绘制曲线；
- 当X是向量，Y是某维数值与X向量相同的矩阵时，MATLAB会绘制多个连续曲线，默认情况下这些曲线的颜色都会不同；曲线的个数等于Y矩阵的另外一个维数，X向量是曲线的横坐标。
- 当X是矩阵，Y是向量时，情况和上面部分相同，Y向量是这些曲线的纵坐标。
- 当X、Y是同维矩阵时，MATLAB会以X、Y对应元素为横、纵坐标绘制曲线，因此，曲线的个数就是矩阵的列数。

◆ plot(X1,Y1, X2,Y2, X3,Y3,...,'PropertyName',PropertyValue,...)

该命令和前面的命令plot(X,Y,'PropertyName',PropertyValue,...)相似，只是同时在图形窗口中绘制多条曲线，这些曲线之间没有相互的约束。

plot 命令的实例

例 6.3 演示下面的案例，来了解上面 plot 的三个调用形式。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> k=0.2:0.2:1;
>>t=0:0.01:1;
>>y=exp(t')*k;
>> subplot(2,2,1),plot(y),axis([ 0,100,0,3] ),title('Picture 1'),xlabel
(' 下标 '),ylabel('y')
>> subplot(2,2,2),plot(t,y),axis([ 0,1,0,3] ),title('Picture 2'),xlabel
('t'),ylabel('y')
>> subplot(2,2,3),plot(y,t),title('Picture 3'),xlabel('y'),ylabel('t')
>> subplot(2,2,4),plot(t),title('Picture 4'),xlabel('下标'),ylabel('t'),
axis([ 0,100,0,1] )
```

step 2 按"Enter"键，就可以得到相应的结果，如图6.3所示。

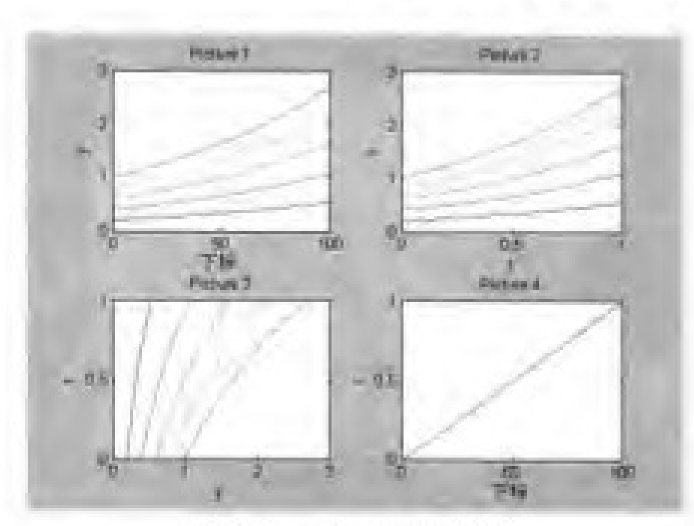


图 6.3 演示 plot 基础命令

step 3 分析图形结果。

在上面的子图 "Picture 1" 中，使用的命令是 "plot(y)"。其中变量 y 是一个 100 × 5 的矩阵，数值就是函数 $y=e^t \times k$ 的计算结果。根据前面的结果，MATLAB 会绘制 5 组（矩阵的列数）曲线，而且曲线以列向量的元素下标为单位，因此，最大下标为 100。从结果中可以看出，这 5 条曲线的颜色是各异的。

在子图 "Picture 2" 中，使用的命令是 "plot(t,y)"。由于 t 是一个 100 维的行向量，y 是一个 100 × 5 的矩阵，因此，MATLAB 会以 t 为横坐标绘制 5 条颜色各异的曲线。

在子图 "Picture 3" 中，使用的命令是 "plot(y,t)"。其情况和上面的子图类似，只是这 5 条曲线都会以 t 为共同的纵坐标。

在子图 "Picture 4" 中，使用的命令是 "plot(t)"。由于 t 是一个 100 维的行向量，因此，MATLAB 会绘制以下标为横坐标的元素曲线。



在上面的图表中，为了显示各个图表坐标系的不同，使用 xlabel、ylabel 命令来添加图表的横坐标和纵坐标。这些命令在编辑图表中也是很常见的，将在后面的章节中详细介绍。

例 6.4 在 MATLAB 中绘制复数矩阵 $[t+e^{-\frac{t}{3}}i, t+\sin(2t+3)i, t+\log(1+t)i]$ 的图形，并在图形中显示对应的图例。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> t=[0:0.1:25]';
>>my_medium=[t,t,t]+i*[exp(-t/3),sin(2*t+3),log(1+t)];
>> plot(my_medium,'Linewidth',2)
>>legend('exp(-t/3)','sin(2*t+3)','log(1+t)')
>> xlabel('t'); ylabel('Y-axis')
```

step 2 按 "Enter" 键，就可以得到相应的结果，如图 6.4 所示。

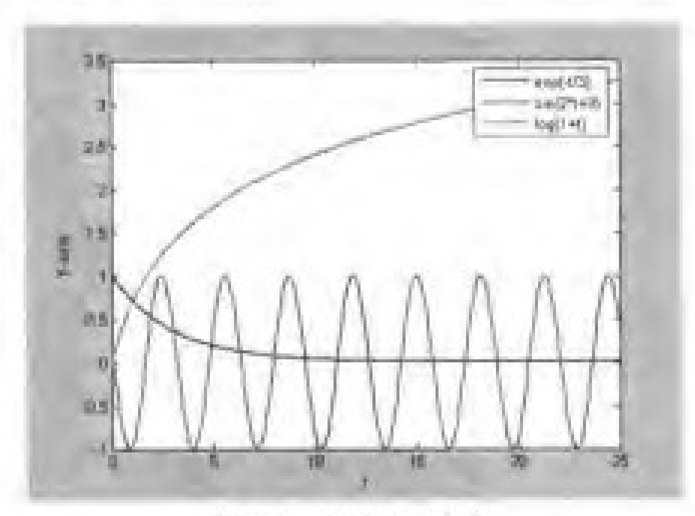


图 6.4　绘制复数矩阵

在上面的图形中，复数矩阵中三个元素分别以实部为横坐标，以虚部为纵坐标，将上面的曲线绘制在同一个图形窗中。为使用户分辨曲线的表达式，使用命令 legend 添加了图表的图例说明。



从上面的步骤中可以看出，在MATLAB的同一个图形窗中绘制不同的曲线命令十分简单，只需要将这些函数表达式写成复数形式，直接使用 plot 命令就可以完成。

6.2.3　曲线的颜色、线型和数据点形

为了能使曲线更加直观，同时也为在复杂图形中分辨各个数据系列，在 MATLAB 中，用户可以为曲线设置不同的颜色、线型和数据点形属性。

在 MATLAB 中，关于曲线的线型和颜色参数的设置如表 6.1 所示。

表 6.1　MATLAB 中线型和颜色的设置

项目	符号	含义
线型	-	实线
	:	虚线
	-.	点划线
	--	双划线
颜色	b	蓝
	g	绿
	r	红
	c	青
	m	品红
	y	黄
	k	黑
	w	白

之所以将曲线的线型和颜色一起介绍，是因为在使用 plot 命令时这两个参数经常一起使用，例如 "r:" 表示曲线是红色的虚线。在命令 plot(X,'PropertyName',PropertyValue,...)中，如果没有输入参数'PropertyName'和 PropertyValue 的数值，MATLAB 会将线型和颜色的属性设置为默认值。线型的默认值是 "实线"；颜色的默认值则依次是蓝、绿、红、青等。如果只有一条曲线，曲线的颜色则是蓝

色的。



上表中列出的是基础颜色，在MATLAB中还可使用RGB颜色系统来为曲线设置颜色。在RGB颜色系统中，颜色是由数组元素数值来控制的，将在后面的章节中详细介绍。

在MATLAB中，用户除了可以为曲线设置颜色、线型之外，还可以为曲线中的数据点设置属性。这样，用户就可以选择不同的数据点形，很方便地将不同的曲线区分开来。MATLAB中的数据点形的属性如表6.2所示。

表6.2 MATLAB中的数据点形

符号	含义	符号	含义
.	实心黑点	+	十字符号
*	八线符	^	上三角符
<	左三角符	>	右三角符
d	菱形	h	六角星
o	空心圆圈	p	五角星
s	方块符	x	叉字符

例6.5 在区间[0,4π]上，绘制函数$y = e^{\frac{x}{3}} \times \sin(3x)$的曲线，同时将在相同的图形窗中绘制该曲线的包络线$y = \pm e^{\frac{x}{3}}$的曲线。对包络线和基本曲线使用不同的线型，并且为曲线上的数据点设置点形。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> x = 0:pi/8:4*pi;
>>y = exp(-x/3).*sin(3*x);
>>yb=exp(-x/3);
>>plot(x,yb,'-k',x,-yb,'-k',x,y,'-.ro','LineWidth',2,...
                  'MarkerEdgeColor','g',...
                  'MarkerFaceColor','y',...
                  'MarkerSize',6)
>>grid on
```

step 2 按"Enter"键，就可以得到相应的图形，如图6.5所示。

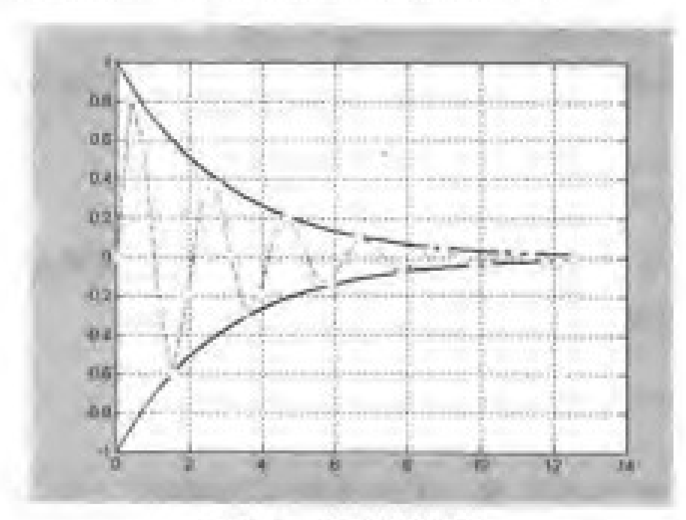


图6.5 设置曲线属性

step 3 分析上面的图形结果。

在上面的图表绘制语句中，只使用一个命令“plot(x,yb,'-k',x,-yb,'-k',x,y,'-.ro','LineWidth',2,'MarkerEdgeColor','g','MarkerFaceColor','y','MarkerSize',6)”就将上面的图表绘制完成。

其中，yb和-yb数据系列是曲线的网络线，其曲线的属性是'-k'，也就是黑色的实线；数组y就是函数的曲线，其曲线的属性是'-.ro'，因此曲线是红色的点划线，同时数据标记是空心圆圈。参数'MarkerEdgeColor'将空心圆圈的边框颜色设置为绿色(g)，参数'MarkerFaceColor'将空心圆圈的填充颜色设置为黄色(y)，参数'MarkerSize'将空心圆圈的大小设置为6。



从上面的例子可以看出，在MATLAB中只需要使用简单的程序语句就可以设置图形的各种属性。灵活使用线型、颜色和数据标记的参数，可以绘制十分美观的图形。

6.2.4 设置坐标轴范围

图表的坐标轴对图表的显示效果有着明显的影响。尽管MATLAB提供考虑比较周全的坐标轴默认设置，但是并不是所有图形的默认效果都是最好的，用户可以根据需要和偏好来设置坐标轴的属性。为此，MATLAB提供一系列关于坐标轴的命令，用户可以根据情况选取合适的命令，调整坐标轴的取向、范围、刻度和高宽比等。表6.3列出了MATLAB中关于坐标轴的常见命令。

表6.3 MATLAB中的数据点形

命令	含义	命令	含义
axis auto	使用默认设置	axis manual	保持当前刻度范围
axis off	取消坐标轴背景	axis on	使用坐标轴背景
axis ij	原点在左上方	axis xy	原点在左下方
axis equal	横、纵坐标使用相同刻度	axis image	等长刻度，坐标框紧贴数据范围
axis normal	默认的矩形坐标系	axis square	正方形坐标系
axis tight	将数据范围设置为刻度	axis fill	使坐标充满整个绘图区

上表中列出的命令中，主要是用于图表坐标轴的控制。在前面的章节中读者已经接触到坐标轴范围的控制命令axis([xmin xmax ymin ymax])，来设置图表各坐标轴的刻度范围。在本小节中，将利用几个例子来说明如何在MATLAB中对图形的坐标轴进行控制。

例6.6 在MATLAB中绘制函数$y=\tan x$在$[0,\frac{\pi}{2}]$中的图表。

step 1 首先使用MATLAB默认的坐标轴范围绘制该图表，在命令窗口输入以下内容：

```
>> x = 0:.01:pi/2;
>>plot(x,tan(x),'-ro')
```

step 2 查看图形结果。按“Enter”键，就可以得到相应的图形，如图6.6所示。

step 3 修改坐标轴刻度范围，在MATLAB的命令窗口输入下列内容：

```
>> axis([0  pi/2  0  50])
>> grid on
```

step 4 查看图形结果。按“Enter”键，就可以得到相应的图形，如图6.7所示。

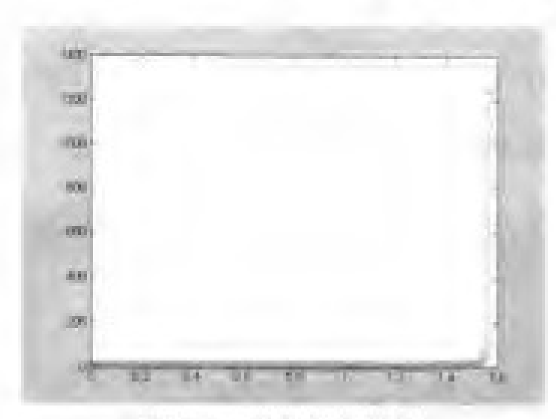


图 6.6 绘制默认图形

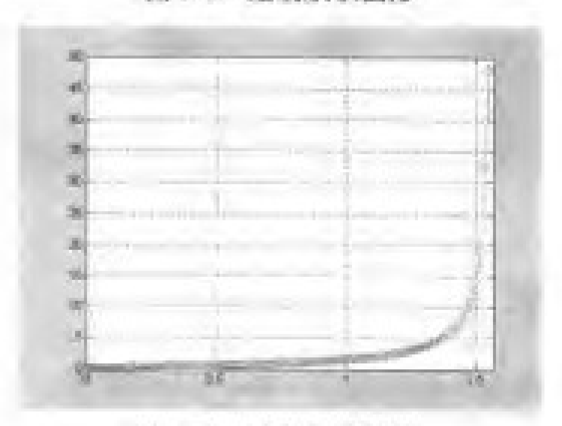


图 6.7 修改后的图形

从上面的两个图形的对比中，发现修改后的图形在描述整个函数性态上要好于默认的图形。这是因为当自变量 x 在整个范围$[0,\frac{\pi}{2}]$中，大部分的函数值都比较小，例如 tan(1.5)=14.1014，tan(1.55)= 48.0785等。但是$\tan(\frac{\pi}{2})=+\infty$，tan(1.57)=1256。也就是说，函数在自变量的很小范围$[1.5,\frac{\pi}{2}]$内递增得十分明显。如果使用 MATLAB 的自动刻度范围，就无法显示整个图表的变化趋势。



在上面的步骤中，之所以将Y轴的最大值设置为50，是因为根据反正切函数，当正切值为50的时候，对应的X数值为1.5508，比较接近pi/2，不会使函数的范围损失太大。

6.2.5 设置坐标轴显示方式

例 6.7 使用不同的坐标轴显示方式，在 MATLAB 中绘制长轴为 4，短轴为 3 的椭圆。

step 1 在命令窗口输入以下内容：

```
>> t=[0:pi/50:2*pi]';
>> x=3*cos(t);
>> y=4*sin(t);
>> subplot(2,3,1),plot(x,y),axis normal,grid on,title('Normal');
>> subplot(2,3,2),plot(x,y),axis equal,grid on,title('Equal');
>> subplot(2,3,3),plot(x,y),axis square,grid on,title('Square');
>> subplot(2,3,4),plot(x,y),axis image,box off,title('Image and Box off');
>> subplot(2,3,5),plot(x,y),axis image fill,box off,title('Image and
```

```
Fill');
>> subplot(2,3,6),plot(x,y),axis tight,box off,title('Tight');
```

step 2 按"Enter"键，就可以得到对应的图形，如图 6.8 所示。

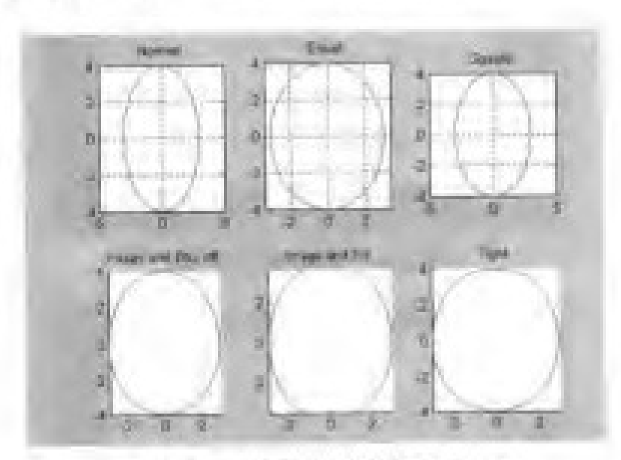


图 6.8　不同的坐标轴显示方式

6.2.6 设置坐标轴系统

例 6.8　在不同的坐标轴系统中绘制函数 $y=\sin(\frac{1}{x})^2/x$ 的图形。

step 1 在命令窗口输入以下内容：

```
>> x=logspace(-2,0,500);
>>y=((sin(1./x)).^2)./x;
>>subplot(2,3,1), plot(x,y); Grid on;title('Normal');
>>subplot(2,3,2), plot(x,y);set(gca,'XScale','log','YScale','linear');
Grid on;Title('Y-Logx');
>>subplot(2,3,3), plot(x,y);set(gca,'XScale','linear','YScale','log');
Grid on;Title('LogY-X');
>>subplot(2,3,4), plot(x,y);set(gca,'XDir','reverse','YDir','normal');
Grid on;Title('Y-revX');
>>subplot(2,3,5), plot(x,y);set(gca,'XDir','normal','YDir','reverse');
Grid on;Title('revY-X');
>>subplot(2,3,6), plot(x,y);set(gca,'XScale','log','YScale','linear');
Title('Y-logX');set(gca,'XGrid','on','YGrid','off')
```

step 2 按"Enter"键，就可以得到对应的图形，如图 6.9 所示。

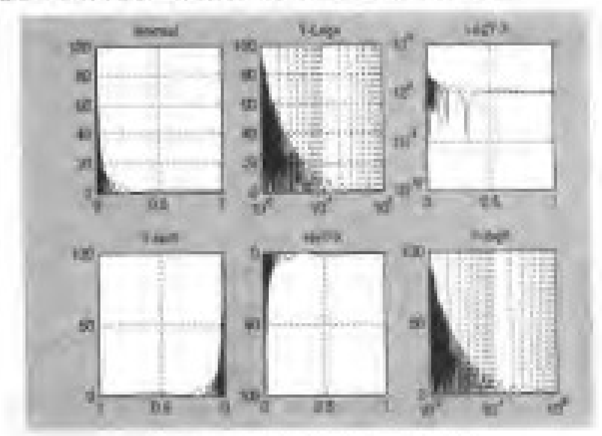


图 6.9　不同的坐标轴系统

step 3 分析图形结果。

上面的图形中总共包括了六个子图形，下面详细介绍这六个图形的内容。

在"Normal"子图形中，使用默认坐标轴系统，也就是变量x、y都在直角坐标系中，而且刻度范围采用MATLAB的默认设置。

在"Y-Logx"子图形中，将*X*坐标轴转化为log*x*的对数坐标系，而*Y*坐标轴保持直角坐标系。由于这个操作涉及坐标轴的转换，因此需要涉及到图表的底层指令。在上面的程序中，使用对象图柄语句set(gca,'XScale','log','YScale','linear')将*X*坐标轴转换为log，而*Y*坐标轴保持默认的linear数值。在"LogY-X"子图形和上面的子图形基本类似，只是将*Y*坐标轴转化为log*y*，而将X坐标轴保持直角坐标系。

在"Y-revX"子图形中，将*X*坐标轴倒置，*Y*坐标轴保持不变。同样，这个操作涉及到了图表坐标轴的转换，因此使用了图表底层的指令：set(gca,'XDir','reverse','YDir','normal')。同理，在"revY-X"子图形中，将*Y*坐标轴倒置，*X*坐标轴保持不变。

在最后一个图表中，使用底层命令set(gca,'XGrid','on','YGrid','off')设置了图表的分格线属性：设置*X*坐标轴的分格线，而将*Y*坐标轴的分格线清除。



在上面的所有图表中对坐标轴的操作都是使用对象图柄命令完成的，因为对于这些操作MATLAB没有提供现成的高层命令。对于这些句柄语句，将在后面的章节中详细介绍其语法。熟悉对象编程的读者可以直接从语句中了解到其作用。

6.2.7 图形标识

在MATLAB中提供多个图形标识命令，来添加多种图形标识。常见的图形标识包括：图形标题、坐标轴名称、图形注释和图例等。关于这些图形标识，MATLAB提供简洁命令方式以及精细命令方式。在本小节中，将使用一个简单的实例来说明图形标识的使用方法。

例 6.9 在MATLAB中绘制函数$f_1(x)=-x^2+4x+450$和$f_2(x)=2^{\sqrt{3x}}+5x$，同时在图形中标识两个曲线的交点和曲线的属性。

step 1 打开M文件编辑器，在其中定义myfun函数，在编辑器中输入以下内容：

```
function f=myfun(x)
y1=-x.^2+4*x+450;
y2=2.^(sqrt(3*x))+5*x;
f=y1-y2
```

之所以要在M文件编辑器中定义该函数，是为了在后面步骤中求解两个曲线的交点，将上面的函数保存为myfun.m文件，然后关闭文件编辑器。



上面的程序语句十分简单，在MATLAB中编写M文件将在后面的章节中详细介绍，本章中需要编写该M文件的原因是后面步骤中需要使用fzero函数来求解曲线的交点。

step 2 在MATLAB的命令窗口中输入以下内容：

```
>>x=[0:0.01:25];
>> y1=-x.^2+4*x+450;
>> y2=2.^(sqrt(3*x))+5*x;
```

```
>> plot(x,y1,x,y2,'Linewidth',2);hold on
>>xrt=fzero(@myfun,18);
>>yrt=-xrt^2+4*xrt+450;
>> plot(xrt,yrt,'r.','Markersize',25);hold off
>> text(xrt,yrt,'\fontsize{16}\leftarrow\fontname{隶书}交点')
>>annotation1 = annotation(...
 'textarrow',...
  [0.6625 0.5036],[0.2262 0.372],...
  'LineWidth',1,...
  'HeadStyle','cback2',...
  'String',{'这是一个','递增函数'},...
  'FontName','IFontb.fon',...
  'FontSize',9,...
  'HorizontalAlignment','left',...
  'TextBackgroundColor',[1 1 1],...
  'TextLineWidth',1,...
  'TextEdgeColor',[0 0 0]);
>>annotation2 = annotation(...
  'textarrow',...
  [0.2732 0.375],[0.5714 0.7143],...
  'String',{'这是一个','递减函数'},...
  'FontSize',9,...
  'TextEdgeColor',[0 0 0]);
>> xlabel('x')
>> ylabel('y')
>> title('两个曲线的交点')
```

step 3 按"Enter"键，就可以得到对应的图形，如图6.10所示。

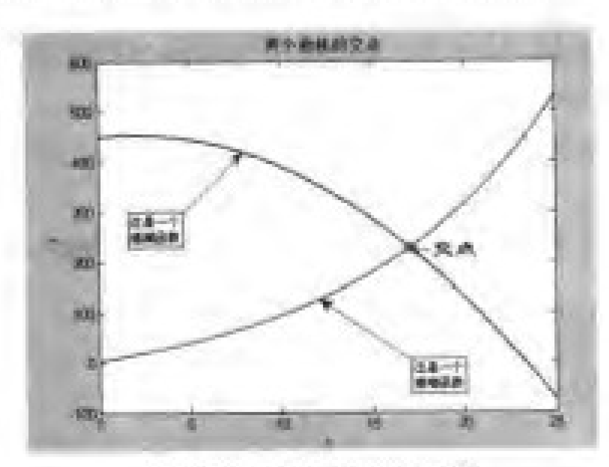


图6.10　完成的图形标识

step 4 分析上面的图形结果。

在上面的图形中，为图形添加了多个图形标识：坐标轴、标题、文本注释、箭头注释等。其中，坐标轴和标题的设置方法十分简单，直接引用简单命令即可，这里就不介绍了。而文本注释和箭头注释相对比较复杂，下面详细分析上面两种注释内容。

命令行"text(xrt,yrt,'\fontsize{16}\leftarrow\fontname{隶书}交点')"为两个曲线交点添加文本注释"交点"。其中参数xrt和yrt表示插入文本注释的位置，也就是坐标点(xrt,yrt)。而参数'\fontsize{16}\leftarrow\fontname{隶书}交点'则定义了注释文字类型（隶书）和大小（16）以及文本内容（交点），其中参数leftarrow表示为文本添加左向箭头。

命令 "annotation1 = annotation('textarrow', [0.6625 0.5036],[0.2262 0.372], 'LineWidth',1, 'HeadStyle','cback2', 'String',{'这是一个','递增函数'}, 'FontName','IFontb.fon', 'FontSize',9, 'HorizontalAlignment','left','TextBackgroundColor',[1 1 1],'TextLineWidth',1, 'TextEdgeColor',[0 0 0]);" 则为图形添加带有箭头的图形注释。其中 "[0.6625 0.5036],[0.2262 0.372]" 表示添加箭头的两个坐标数值，后面的参数分别设置了图形注释文本的属性。



在上面的程序语句中的fzero函数，可以使用MATLAB的帮助文件来查阅相应的用法。该函数的功能是求解某公式的零点。

6.2.8 叠绘

在MATLAB中，用户可以使用多种命令在同一个图形窗中绘制不同图形；但是在实际应用中，用户会遇到在已经完成的图形中再绘制曲线的情况。为了达到这种效果，MATLAB为用户提供了hold命令。

在MATLAB中，hold命令的常用调用格式为：

- hold on 启动图形保持功能，当前坐标轴和图形都将保持，此后绘制的图形都将添加在这个图形基础上，并且自动调整坐标轴的范围；
- hold off 关闭图形保持功能；
- hold 在hold on 和hold off命令之间进行切换。

例 6.10 在MATLAB中依次绘制函数 $f_1(x)=-x^2+4x+450$ 和 $f_2(x)=2^{\sqrt{3x}}+5x$ 的曲线，然后再将 $f_2(x)$ 向下平移50个单位。

step 1 在MATLAB 的命令窗口中输入下面的内容：

```
>> x=[0:0.01:25];
>> y=-x.^2+4*x+450;
>> plot(x,y,'LineWidth',2);hold on
```

step 2 按 "Enter" 键，得到第一个函数的曲线，如图6.11所示。

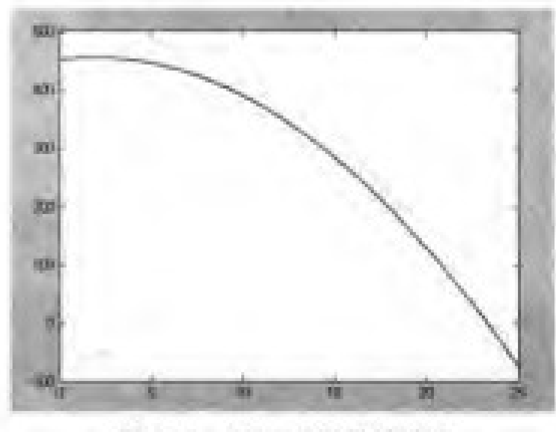


图 6.11 第一个函数的曲线

step 3 在MATLAB 的命令窗口中输入下面的内容：

```
>> f=2.^(sqrt(3*x))+5*x;
>> plot(x,f,'LineWidth',2);
```

step 4 查看图形结果。输入代码之后，按“Enter”键，得到的曲线如图 6.12 所示。

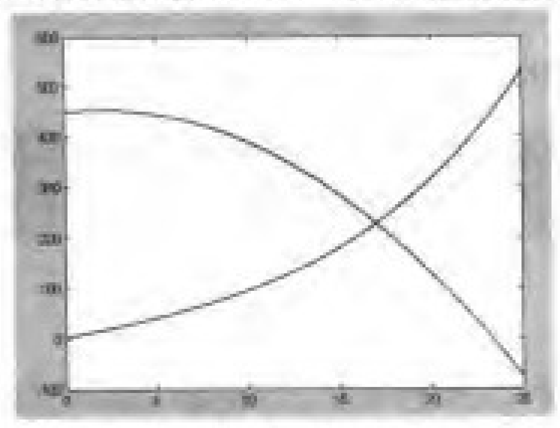


图 6.12　添加第二个函数的曲线

step 5 在 MATLAB 的命令窗口中输入下面的内容：

```
>> plot(x,f-50,'LineWidth',2);
>> hold off;
```

step 6 查看图形结果。输入代码之后，按“Enter”键，得到的曲线如图 6.13 所示。

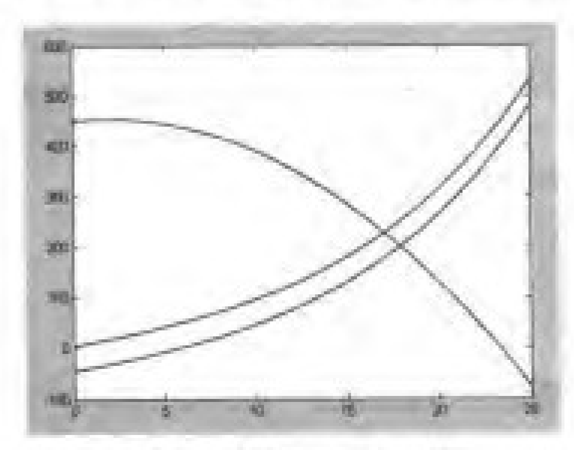


图 6.13　添加第三个函数的曲线

step 7 最后，在 MATLAB 的命令窗口中输入下面的内容：

```
>> plot(x,f-50,'LineWidth',2);
```

step 8 查看图形结果。输入代码之后，按“Enter”键，得到的曲线如图 6.14 所示。



上面的步骤并不复杂，但能比较形象地说明 hold 命令的作用。在一个函数曲线绘制完成后，使用 hold on 命令，开启图形保持功能，因此在图形中添加第二个函数曲线的时候，第一个函数的曲线会保留在窗口中。由于用户没有关闭图形保持功能，因此可以直接在这个基础上添加第三个函数的曲线；而当用户使用 hold off 命令关闭图形保持功能后，再次绘制函数曲线时，图形窗口中就只有最后一个函数的曲线。

图 6.14 单独绘制最后一个函数的曲线

6.2.9 双坐标轴

双坐标轴问题是科学计算和绘图中经常遇到的问题，当需要将同一自变量的两个（或者多个）不同量纲、不同数量级的函数曲线绘制在同一个图形中时，就需要在图形中使用双坐标轴。为了满足这种需求，MATLAB 提供 plotyy 命令。

plotyy 命令的常用调用格式如下：

- plotyy(x1,y1,x2,y2) 以左、右不同纵轴绘制 x1-y1、x2-y2 两条曲线；
- plotyy(x1,y1,x2,y2,FUN) 以左、右不同纵轴绘制 x1-y1、x2-y2 两条曲线，曲线类型由 FUN 指定；
- plotyy(x1,y1,x2,y2,FUN1,FUN2) 以左、右不同纵轴绘制 x1-y1、x2-y2 两条曲线，x1-y1 曲线类型由 FUN1 指定，x2-y2 曲线类型由 FUN2 指定。



在上面的命令格式中，左坐标轴用于绘制 x1-y1 数据系列，右坐标轴用于绘制 x2-y2 数据系列。坐标轴的范围和刻度都可以自动产生。参数 FUN、FUN1、FUN2 可以是 MATLAB 中所有可以接受的二维绘图命令，例如 plot、bar、linear 等。

例 6.11 某公司统计了公司近半年的销售收入（Income）和边际利润率（Profit_Margin）的数据，为了方便财务人员进行查看，需要在同一个图形窗中绘制两组数据的变化趋势。需要在MATLAB 中实现上面的要求。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> Income=[ 2456,2032,1900,2450,2890,2280] ;
>> Profit_Margin=[ 12.5,11.3,10.2,14.5,14.3,15.1] /100;
>> t=1:6;
>> [ AX,H1,H2] = plotyy(t,Income,t,Profit_Margin,'bar','plot');
```

step 2 查看图形结果。输入代码之后，按 "Enter" 键，得到的曲线如图 6.15 所示。

step 3 然后在 MATLAB 的命令窗口中输入下面的内容：

```
>>set(get(AX(1),'Ylabel'),'String','Left Y-axis');
>>set(get(AX(2),'Ylabel'),'String','Right Y-axis');
 >>set(H2,'LineWidth',2,...
```

```
    'Marker','o',...
    'MarkerEdgeColor',[1 0 0],...
    'MarkerFaceColor',[1 0 0]);
>>set(H1,'BarWidth',0.6,...
    'EdgeColor',[1 0 0],...
    'FaceColor',[1 1 0.4745]);
```

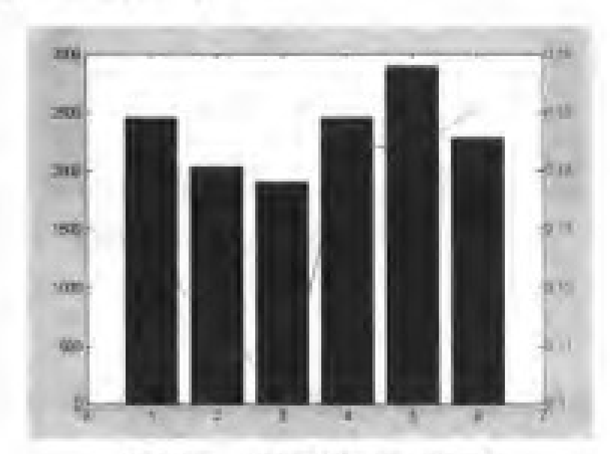


图 6.15　绘制的基础双轴图形

step 4 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.16 所示。

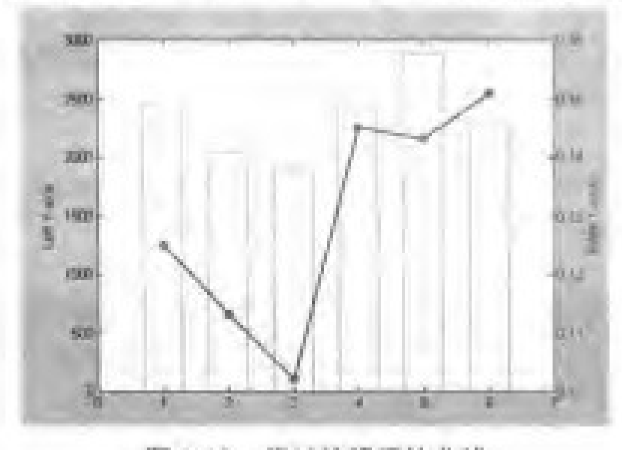


图 6.16　经过编辑后的曲线

step 5 分析上面的图形结果。

在上面的步骤（1）中，使用plotyy函数绘制了基础的双轴曲线，对于第一个数据系列Income使用的图表类型是直方图（bar），对于第二个数据系列Profit_Margin使用的图表类型是直线（plot）。在没有任何编辑之前，MATLAB会将这两条曲线的属性都设置为默认值，例如两个曲线的颜色都是蓝色等。

在步骤（2）中，通过使用底层的图型句柄语句来设置两个曲线的属性，例如线宽、填充颜色、数据标记等。这些语句的功能主要是编辑图形的外观，让图形的对比更加明显。这些语句都涉及图形句柄的内容，这些内容将在后面的章节中加以详细介绍。



> 如果需要在双轴图表图形中添加文字标记，那么text命令添加的标记位置由左纵轴决定，同时，前面介绍的图例命令legend在双轴图形中无法正常使用。

6.2.10 多子图

在前面小节中，已经多次遇到在一个图形窗口中布置多个独立的子窗口的情况。在同一图形窗中绘制多个子图，可以很方便地将几个独立图形进行对比。在 MATLAB 中，提供 subplot 命令来绘制子图。

subplot 命令的常见调用格式如下：

- ubplot(m,n,p) 该命令的功能是将图形窗口分成 m × n 个子窗口，然后在第 p 个小窗口中创建坐标轴，将其设置为当前窗口；
- subplot(m,n,p,'replace') 如果在绘制图形的时候已经定义了坐标轴，该命令将删除原来的坐标轴，创建一个新的坐标轴系统；
- subplot(m,n,p,'align') 该命令的功能是对齐坐标轴；
- subplot(h) 使句柄 h 对应的坐标轴成为当前坐标轴；
- subplot('Position',[left bottom width height]) 在指定位置上开辟子图，并将其设置成当前坐标轴。



在命令 subplot(m,n,p)中，p 代表的是子图的编号。子图序号编排规则是：左上方是第 1 幅，然后向右向下依次编号，这个命令产生的子图分割按照默认值设置。而在命令 subplot('Position',[left bottom width height])中，指定位置的四元数组必须采用标化坐标，图形窗的高和宽分别是[0,1]，左下方的坐标是(0,0)。最后，所有的 subplot 命令产生的子图都是相互独立的。

例 6.12 在 MATLAB 中，用子图绘制函数 $y_1 = e^{-\frac{t}{3}}, y_2 = \sin(2x+3), y_3 = \log(1+x)$ 的图形，分别使用不同的子图绘制命令。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> t=[ 0:0.1:25] ';
>> y1=exp(-t/3);y2=log(1+t);y3=sin(2*t+3);
>> subplot(2,2,1),plot(t,y1,'Linewidth',2)
>> subplot(2,2,2),plot(t,y2,'Linewidth',2)
>> subplot('position',[ 0.2,0.05,0.6,0.45] )
>>  plot(t,y3,'r','Linewidth',2)
```

step 2 按 "Enter" 键，结束输入并执行命令，得到的结果如图 6.17 所示。

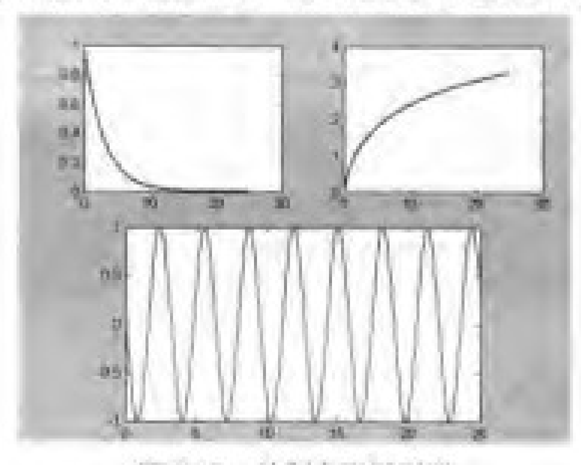


图 6.17 绘制多子图图形



在上面的命令中，如果最后一个子图的绘制命令中将子图的高度增加，例如改为0.7，那么最后一个子图会将上面所有的子图覆盖。

6.2.11 交互式图形命令

在MATLAB中，除前面介绍的图形编辑命令之外，还提供一些和鼠标操作相关的命令，也就是交互式图形命令。例如，当用户将某个函数的曲线绘制完成以后，有时候希望知道某个自变量数值下的函数值，这个时候使用ginput命令可以十分便利地通过鼠标来读取二维平面图形中的任意一点的坐标值。

除了ginput命令之外，MATLAB还提供gtext、zoom等命令，这些命令都和鼠标操作有关。在这些命令中，除了ginput命令只能用于二维图形之外，其他的命令都可以用在二维和三维图形中。其中ginput命令和zoom命令经常一起使用，可以从图形中获得比较准确的数据。

在MATLAB中，ginput命令的常用调用格式如下：

- [x,y] = ginput(n) 该命令的功能是用鼠标从二维图形上获取n的数据点的坐标数值，用Enter键来结束取点；
- [x,y] = ginput 取点的数目不受限制，这些数据点的坐标数值保存在[x,y]中，通过Enter键来结束取点；
- [x,y,button] = ginput(...) 返回值button记录了在选取每个数据点的相关信息。



这个命令和其他图形命令的机理不同，其他命令是将各个数据点绘制到图形中，而该命令则从已经绘制完成的图形中获取数据点的信息。同时，该命令只适合用于二维图形。命令中的n只能选取正整数，表示选择数据点的个数。

在MATLAB中，ginput命令经常和zoom命令一起配合使用。zoom命令的功能在于实现对二维图形的缩放，其常见的调用格式如表6.3所示。

表6.3 MATLAB中的zoom命令

命令	含义	命令	含义
zoom	进行命令切换	zoom factor	将factor作为缩放因子进行坐标轴的缩放
zoom on	允许对坐标轴进行缩放	zoom off	取消对坐标轴进行缩放
zoom out	恢复对坐标轴的设置	zoom reset	将当前的坐标轴设置为初始值
zoom xon	允许对X轴进行缩放	zoom yon	允许对y轴进行缩放



在MATLAB中，使用zoom命令完成的缩放工作是标准的Windows操作：单击左键，就可以将图形放大，也可以使用鼠标拉出一定的区域后将区域进行放大；单击右键完成的则是相反的操作，也就是将图形缩小。默认的缩放因子是2，用户可以使用factor来修改这个缩放因子。

6.2.12 交互式图形实例

例6.13 在MATLAB图形窗口中实现动态绘图：使用鼠标左键选取曲线的数据点，用鼠标右键

结束，然后使用曲线将上面的所有数据点连接起来，绘制曲线。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> axis([ 0 10 0 10] )
>> hold on
>> xy = [] ;
>> n = 0;
>> disp('Left mouse button picks points.')
>> disp('Right mouse button picks last point.')
>> but = 1;
>> while but == 1
    [ xi,yi,but] = ginput(1);
    plot(xi,yi,'ro')
    n = n+1;
    xy(:,n) = [ xi;yi] ;
end
>> t = 1:n;
>> ts = 1: 0.1: n;
>> xys = spline(t,xy,ts);
>> plot(xys(1,:),xys(2,:),'b-','linewidth',2);
>> hold off
```

step 2 查看图形结果。输入代码之后，按"Enter"键，MATLAB会弹出图形窗，使用鼠标左键选择数据点，然后点击鼠标右键，结束数据点选取，得到的图形如图6.18所示。

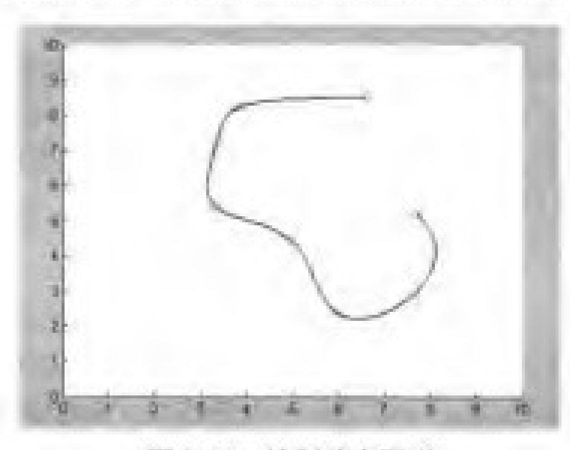


图 6.18　绘制动态图形

step 3 重新选择数据，绘制新图形。上面的图形程序对数据点的选取次序、个数没有任何限制，用户可以根据需要随意选取。因此，可以重新选择数据点，然后使用鼠标完成下面的图形，如图6.19所示。



在上面的程序中，使用了循环语句来依次将用户使用鼠标选取的数据点信息添加到对应的数组中，然后使用样条函数对选取的数据点进行插值。在程序的最后，使用plot命令绘制插值后的曲线。关于插值和程序的相关知识，请参阅本书中相应的章节。

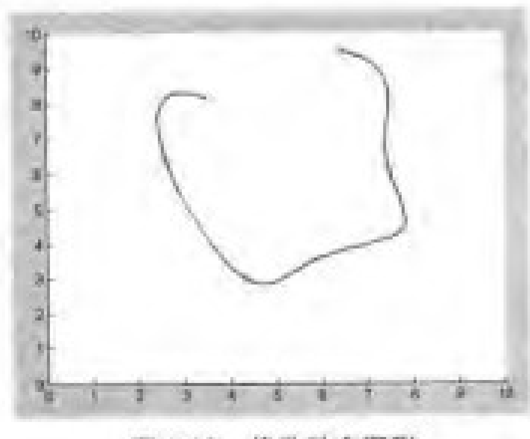


图 6.19　修改动态图形

6.2.13　fplot 命令

在前面的小节中，读者已经对plot命令有了详细的了解。其实MATLAB对于图形不同的数据来源提供了不同的绘图命令。其中比较常见的命令是fplot和ezplot命令，两种命令的适用范围互不相同。

简单来说，前面小节介绍的plot命令是将函数数值得到的数值矩阵转化为连线图形。在实际应用中，如果不太了解某个函数随自变量变化的趋势，而使用plot命令绘制该图形时，就有可能因为自变量的范围选取不当而使函数图形失真。可以根据微分的思想，将图形的自变量间隔取得足够小来减小误差，但是这种方法会增加MATLAB处理数据的负担，降低效率。

对于上面的问题，MATLAB提供fplot函数来解决。该命令用来指导如何通过函数取得绘图的数值数据矩阵，命令通过内部的自适应算法来动态决定自变量的间隔：当函数值变化比较缓慢时，自变量的间隔取得大一点；当函数数值变化比较剧烈时，自变量的间隔就会取得小一点。这样，就可以十分方便地保证绘图的质量和效率。

在MATLAB中，fplot命令的常用调用格式如下。

- fplot(function,limits,tol,LineSpec)　在这个命令中，参数function表示需要绘制曲线的函数名称；limits表示绘制图形的坐标轴取值范围，可以有两种方式：[xmin xmax]表示图形X坐标轴的取值范围，[xmin xmax ymin ymax]则表示X、Y坐标轴的取值范围；tol表示函数相对误差容忍度，默认值为2e-3；LineSpec表示图形的线型、颜色和数据点等。
- [...] = fplot(function,limits,tol,n,LineSpec,P1,P2,...)　在这个命令中，fplot函数通过相关参数计算取值后，向函数传递P1,P2,...等函数参数数值。

6.2.14　使用fplot命令绘制图形

例 6.14　在MATLAB中绘制函数 $y_1 = 200\frac{\sin x}{x}$ 和 $y_2 = x^2$ 的图形。



上面两个函数的表达式并不复杂，但是两个函数有着明显的区别。y1的函数在x=0附近变化较大，如果希望很好地表现函数变化，需要在零点附近将自变量的间隔减少；函数y2在整个取值范围之内变化都比较平缓，因此需要使用fplot来解决这个问题。

step 1 打开 MATLAB 的 M 文件编辑器，在其中输入下面的函数表达式：

```
function Y = fplot_fun(x)
Y(:,1) = 200*sin(x(:))./x(:);
Y(:,2) = x(:).^2;
```

然后将其保存为 fplot_fun.m，这个文件就是后面步骤中需要绘制的函数表达式。

step 2 在 MATLAB 的命令窗口中输入下面的内容：

```
>> fh = @fplot_fun;
>> [X,Y]=fplot(fh,[-20 20],2e-4);
>> L=size(X);
>> px=-20:40/(L(1)+1):20;
>> py=fplot_fun(px);
>> subplot(2,1,1),plot(X,Y,'Linewidth',2),title('fplot')
>> subplot(2,1,2),plot(px,py,'Linewidth',2),title('plot')
```

step 3 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.20 所示。

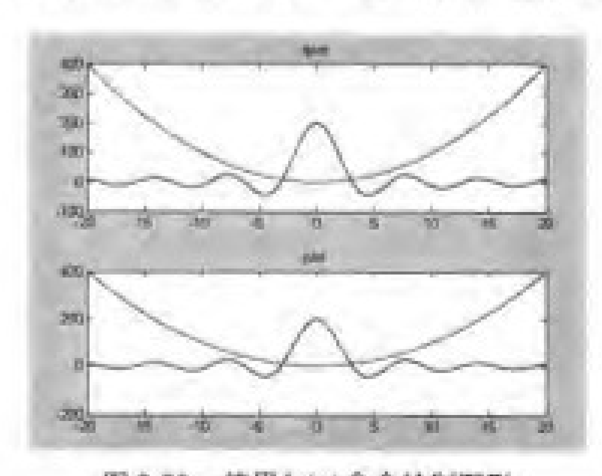


图 6.20 使用 fplot 命令绘制图形



在上面的步骤中，首先使用 fplot 命令绘制两个函数的图形，然后定义自变量的取值范围后，再使用 plot 命令绘制函数的图形。图形在外观上是完全一致的，但是在处理效率上是有很大区别的。

6.2.15 ezplot 命令

ezplot 命令是 MATLAB 为用户提供的简易二维图形命令。其命令名称的前两个字符"ez"的含义就是"Easy to"，表示对应的命令是简易命令。这个命令的最大特点就是，不需要用户对图形准备任何的数据，就可以直接画出字符串函数或者符号函数的图形。

在 MATLAB 中，ezplot 命令的常用调用格式如下：

- ezplot(f) 在默认的自变量范围 $[-2\pi,2\pi]$ 内绘制函数 f 的曲线；
- ezplot(f,[min,max]) 在用户自定义的自变量范围内绘制 f 的曲线；
- ezplot(f,[min,max],fig) 在制定的图形窗口中，在自定义的自变量范围内，绘制函数 f 的曲线。



在上面的函数中，参数f可以是字符表达函数、符号函数、内联函数等，但是所有的函数类型只能是一元函数。在默认情况下，ezplot命令会将函数表达式和自变量写成图形名称和横坐标名称，用户可以根据需要使用title、xlabel命令来命名图名称和横坐标名称。

6.2.16 使用 ezplot 命令绘制图形

例 6.15 在 MATLAB 中绘制函数 $y=\frac{3}{4}e^{-\frac{2t}{3}}\sin(1+2t)$ 的图形。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> syms t
>>y=3/4*exp(-2*t/3)*sin(1+2*t);
>>ezplot(y,[pi,3*pi]);grid
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.21 所示。

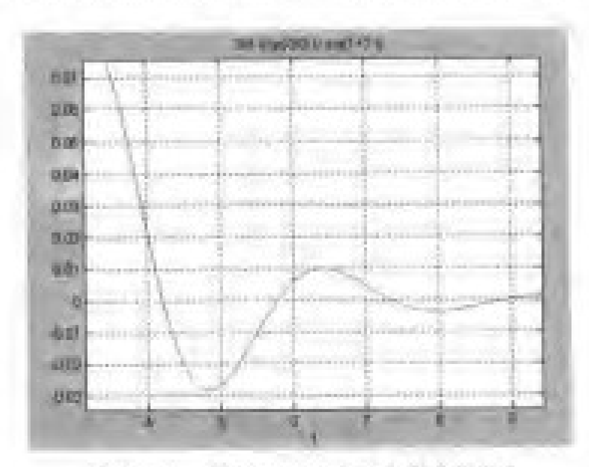


图 6.21 使用 ezplot 命令绘制图形

从上面的图形中可以看出，首先使用 syms 命令将 t 设置为符号变量，然后输入符号表达式，最后使用 ezplot 命令来绘制图形。用户并没有为函数准备任何的自变量的数据，MATLAB 自动为其选定自变量范围，同时计算相应的函数数值。



在 ezplot 命令中，不能使用前面章节中 plot 命令中的参数为曲线指定曲线线型、色彩等，也不允许同时在一个图形窗口中绘制多个图形。但是，前面章节中介绍的 text、grid、zoom 等命令还是可以适用于 ezplot 命令的。

6.3 绘制三维曲线

从本节开始，将详细介绍在 MATLAB 中绘制三维曲线的命令和方法。尽管二维绘图和三维绘图在很多地方是共通的，但是三维曲线在很多方面是二维图形曲线没有涉及的，因此，本节需要详细介绍三维曲线的命令方法。

6.3.1 plot3 命令

和二维曲线命令相似，plot3命令是绘制三维曲线的基础命令，也是最简单的命令，其调用格式和plot非常相似。其具体的调用格式如下：

plot3(X,Y,Z,LineSpec, 'PropertyName',PropertyValue,...) 对于不同情况下的输入参数X、Y和Z，该命令会得出不同的曲线结果。

- 当X、Y和Z是同维向量时，MATLAB会绘制以X、Y和Z元素为x、y、z坐标的三维曲线。
- 当X、Y和Z是同维矩阵时，MATLAB会绘制以X、Y和Z对应列元素为x、y、z坐标的三维曲线，曲线的个数等于矩阵的列数。
- 在上面的命令中，参数LineSpec主要是定义曲线的线型、颜色和数据点等，参数PropertyName是曲线对象的属性名，PropertyValue是对应属性的取值。

6.3.2 plot3 命令实例

例 6.16 在MATLAB中绘制圆锥螺线的三维图形。

根据高等数学知识，圆锥螺线的三维参数方程如下：

$$\begin{cases} x = vt\sin\alpha\cos wt \\ y = vt\sin\alpha\sin wt \\ z = vt\cos\alpha \end{cases}$$

在上面的参数方程中，圆锥角为2α，旋转角速度为ω，直线速度为v。为了简化绘制过程，在本实例中仅保留参数t来绘制三维曲线，具体步骤如下：

step 1 在MATLAB的命令窗口中输入下面的内容：

```
>> v=20;
>>alpha=pi/6;
>>omega=pi/6;
>> x=v*sin(alpha)*t.*cos(omega*t);
>>y=v*sin(alpha)*t.*sin(omega*t);
>>z=v*cos(alpha)*t;t=[0:0.01*pi:50*pi];
>>plot3(x,y,z,'r','LineWidth',2)
>>grid on
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图6.22所示。

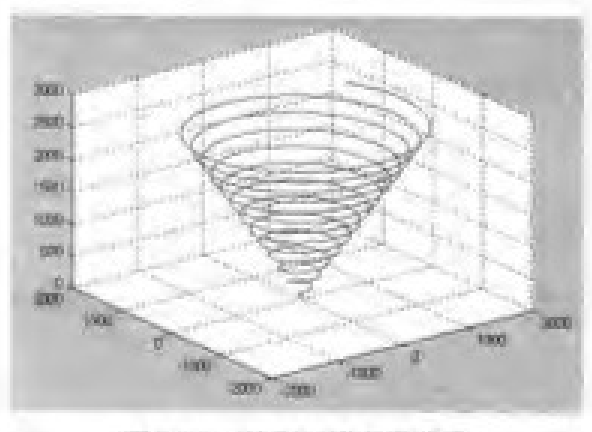


图 6.22 绘制三维圆锥曲线



在 MATLAB 中，三维曲线的绘制命令 plot3 主要用来绘制单参数的三维曲线；对于有多个参数的曲线，需要使用其他的绘图命令。

6.3.3 曲线图命令

当绘制三维曲线的时候，除了需要绘制单根曲线，还经常需要绘制三维曲线的曲线图和曲面图。为此，MATLAB 提供 mesh 和 surf 命令，分别用来绘制三维曲线图和曲面图。

在 MATLAB 中，绘制三维函数 z=f(x,y)的时候进行数据准备的原理基本上等于下面的程序代码：

```
>> x=x1:dx:x2;y=(y1:dy:y2)';
>> X=ones(size(y))*x;
>> Y=y*ones(size(x));
```

在上面的程序代码中，得到的X、Y数组就是x-y平面上的自变量采样"格点"矩阵。其中，dx、dy表示两个自变量的取值间隔，而X、Y则使用相应的命令产生数据点的矩阵。MATLAB会以此数据点矩阵为自变量取值，绘制相应的网格线。

在 MATLAB 中，mesh 的常用调用方式如下：

- mesh(z)　以 z 为矩阵列、行下标为 x、y 轴自变量，绘制网线图；
- mesh(x,y,z)　最常用的网格线调用格式；
- mesh(x,y,z,c) 最完整的调用方式，画出由 C 指定的用色的网格线。

上面的调用格式中，相同参数的含义是一致的，因此在这里仅仅解释最完整的调用方式中各个参数的含义。其中，x、y是自变量"格点"矩阵；z是建立在"格点"之上的函数矩阵；C是指定各点用色的矩阵，一般情况下可以省略。默认情况下，如果用户没有指定矩阵 C，MATLAB 会认为 C=Z。

6.3.4 曲线图实例

例 6.17　在 MATLAB 中绘制函数 Peaks 的三维网格线。



Peaks 是 MATLAB 的内置函数，是一个二元函数，作用是返回 Gaussian（高斯）分布的数值。默认情况下，Peaks 的参数 x 和 y 的取值范围都是[−3,3]。函数 Peaks(N)将会返回一个 $N\times N$ 矩阵，该函数在显示三维图形中有着广泛的应用；本小节为了简化步骤，可以绘制 Peaks(25)的曲线。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>>z=peaks(25);
>>mesh(z)
>> colormap(cool)
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.23 所示。

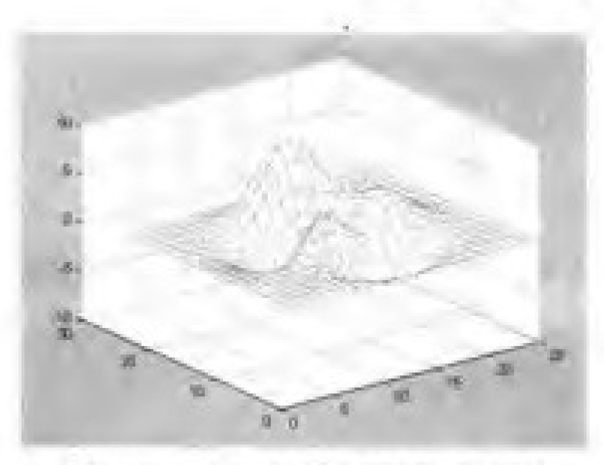


图 6.23 使用 mesh 命令绘制三维曲线

在上面的程序代码中，使用了colormap函数来为三维图形设置颜色。该命令是MATLAB的内置函数，可以使用RGB和HSV系统对曲线进行着色。在图形窗口中选择 "Edit" ⇨ "Colormap" 命令，打开 "Colormap Editor" 对话框，在该对话框中对图形的颜色进行更加详细的设置，如图 6.24 所示。

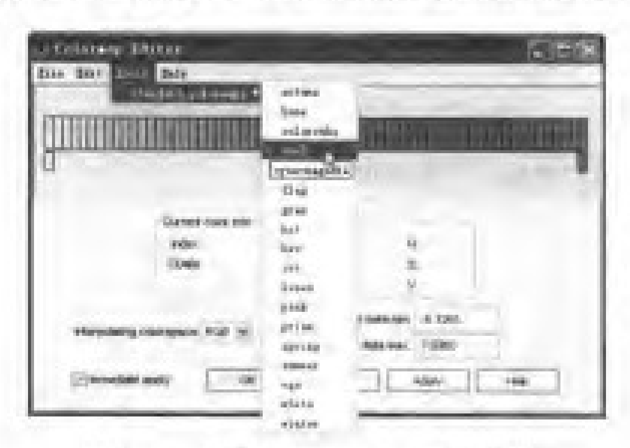


图 6.24 "Colormap Editor" 对话框

在 "Colormap Editor" 对话框中，可以选择 "Tool" ⇨ "Standard colormap" 命令，查看colormap命令的常见参数，例如hot、jet、bone等。这些参数其实是MATLAB内部程序设定的颜色参数，例如，cool代表的是 "cyan-magenta" 颜色，也就是本实例中的颜色。



colormap函数经常被用来为三维曲线设置颜色，了解该函数的参数会使三维曲线的颜色设置十分方便。可以在MATLAB中查看关于该函数的帮助信息。

6.3.5 曲线图和等高线命令

在MATLAB中，还有很多和mesh命令相互联系的命令，例如meshz、meshc等。这些命令的调用格式都和mesh命令相似，只是在功能上有些区别，主要区别如下：

- meshc 的功能是在 mesh 命令绘制的三维曲面图下绘出等高线；
- meshz 的功能是在 mesh 命令的作用之上增加绘制边界的功能。

例 6.18 在 MATLAB 中绘制函数 Peaks 的三维曲面的等高线和边界曲线。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> z=peaks(25);
>>meshc(z)
>>colormap(hsv)
```

step 2 查看图形结果。输入代码之后，按 "Enter" 键，得到的曲线如图 6.25 所示。

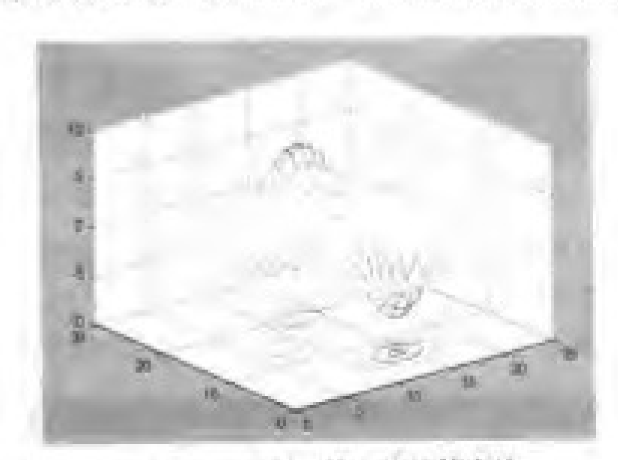


图 6.25　绘制三维曲线的等高线

step 3 在 MATLAB 的命令窗口中输入下面的内容：

```
>> clf
>> z=peaks(25);
>> meshz(z)
>> colormap(jet)
```

step 4 查看图形结果。输入代码之后，按 "Enter" 键，得到的曲线如图 6.26 所示。

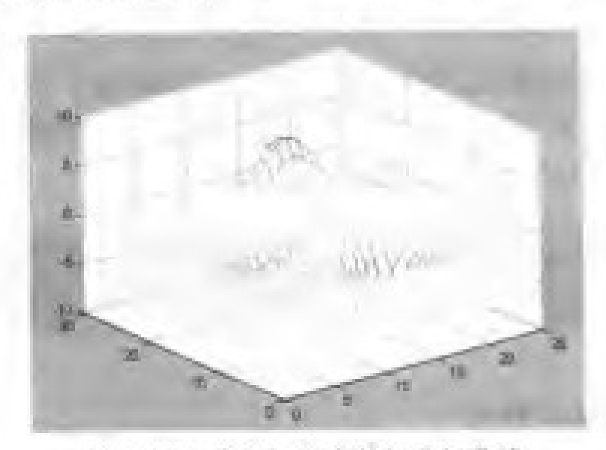


图 6.26　绘制三维曲线的边界曲线

6.3.6 曲面图命令

在 MATLAB 中，绘制三维曲面图的主要命令是 surf，该命令可以绘制已经着色的三维曲面。默认的着色方法是得到相应的网格后，对每一个网格依据该网格所代表的节点的色值来定义该网格的颜色。

在 MATLAB 中，surf 命令的常见调用格式如下：

- surf(z)　以 z 为矩阵的列、行下标为 x、y 轴的自变量，绘制曲面图；

- surf(x,y,z) 最常用的曲面图线调用格式；
- surf(x,y,z,c) 最完整的调用方式，画出由 C 指定的用色的曲面图。

上面调用命令中的参数和前面介绍的 mesh 命令类似，这里就不再重复介绍了。

6.3.7 曲面图实例

例 6.19 在 MATLAB 中绘制函数 Peaks 的三维曲面图。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> z=peaks(25);
>>surf(z)
>>colormap(hsv)
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.27 所示。

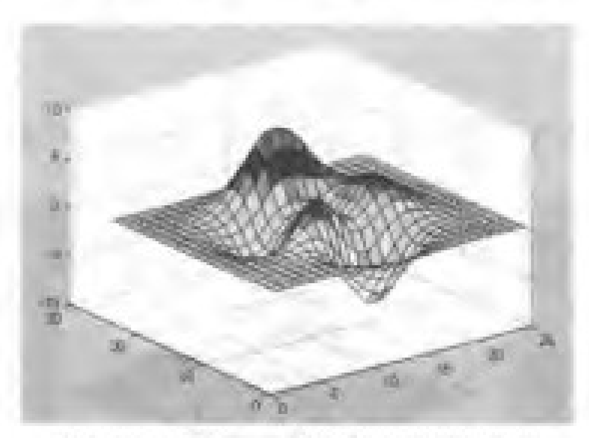


图 6.27 使用 surf 命令绘制三维曲面图

在 mesh 命令类似，在 MATLAB 中也有一些和 surf 命令相互联系的命令，例如 surfl、surfc 等。这些命令主要功能如下：

- surfl 的功能是产生一个带有阴影效果的曲面，该阴影效果是基于环绕、分散、特殊光照等模型得到的；
- surfc 的功能是在 surf 命令的作用之上增加曲面的等高线。

例 6.20 在 MATLAB 中绘制函数 Peaks 的三维曲面图的阴影曲面和等高线。

step 1 在 MATLAB 的命令窗口中输入下面的内容：

```
>> z=peaks(25);
>> surfl(z)
>>shading interp
>>colormap copper
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的图形如图 6.28 所示。

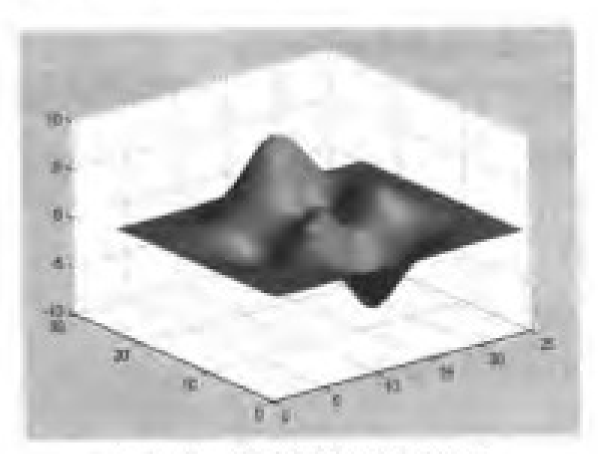


图 6.28　绘制阴影效果的曲面

step 3 在 MATLAB 的命令窗口中输入下面的内容：

```
>> z=peaks(25);
>> surfc(z)
>> colormap hsv
```

step 4 查看图形结果。输入代码之后，按"Enter"键，得到的图形如图 6.29 所示。

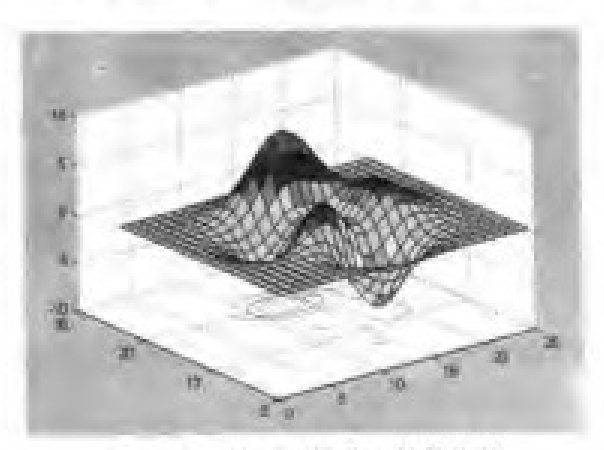


图 6.29　绘制三维曲面的等高线

6.4 特殊图形

在 MATLAB 中，为用户提供了不少特殊图形的命令，使用这些绘图命令可以很方便地绘制出一些特殊图形：面积图、直方图、矢量图等。这些图形使用基础的 plot 命令也是可以完成的，但是操作步骤比较复杂，而使用 MATLAB 内部已经设置好的命令就可以很方便地完成这些特殊图形。

本节将选取 MATLAB 中一些典型的二维和三维特殊图形的命令，介绍在 MATLAB 中绘制图形的方法。之所以将二维和三维曲线一起介绍，是因为这些特殊图形的命令有很多是二维和三维通用的。

6.4.1 area 命令

在 MATLAB 中，为用户提供了绘制面积图的命令 area，这个命令适用于绘制二维面积图。这个命令的主要特点是：在图形中绘制多条曲线时，每条曲线都将前面的曲线当做基线，然后取值绘制曲线。

area 命令的常见调用格式如下：

- area(Y) 这是比较常见的调用格式，将以向量 Y 的下标为横坐标轴，以向量 Y 的元素数值为纵坐标轴来绘制面积图；
- area(X,Y) 当 X 和 Y 都是向量的时候，这个命令绘图的结果和 plot(X,Y)相同，只是在默认的 X 轴最小值 0 和数值 Y 之间有填充效果；当 X 是向量、Y 是矩阵的时候，则以向量 X 为横坐标轴，以矩阵的列元素的累积数值为纵坐标数值绘制填充图形；
- area(...,basevalue) 在这个命令中，为面积图设置了填充的基值数值，如果没有指定该参数的数值，MATLAB 会将该数值设置为 0。



对于 area 命令绘制的基础图形，前面小节介绍的图形编辑命令都可以应用，例如 grid、colormap 等命令都可以使用在该命令绘制的图形中。

6.4.2 面积图实例

例 6.21 在 MATLAB 中绘制面积图。

step 1 在 MATLAB 中的命令窗口中输入下面的命令：

```
>> Y = [2, 4, 5;
3, 1, 9;
5, 3, 5;
2, 6, 1];
>>area(Y,'LineWidth',2)
>>grid on
>>colormap summer
>>set(gca,'Layer','top')
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的图形如图 6.30 所示。

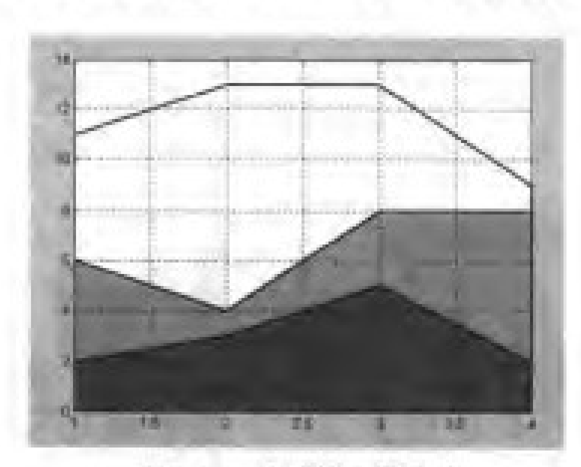


图 6.30 完成的面积图形



在上面的程序代码中，使用了高级图形句柄语句set(gca,'Layer','top')将面积图的图层设置为最顶层，这样图形的分格线就可以显示在其上面。

6.4.3 bar 命令

在 MATLAB 中，提供命令 bar 和 bar3 来绘制二维和三维垂直条形图，提供命令 barh 和 bar3h 来绘

制二维和三维水平条形图。两种绘制命令有很多相似的地方，但是也有一些不同的地方。

在 MATLAB 中，命令 bar 的常用调用方式如下：

- bar(Y) 为向量 Y 中的每一个元素绘制一个条形。
- bar(x,Y) 在指定的横坐标轴 x 上绘制 Y，其中参数 x 必是严格递增的向量。
- bar(...,width) 参数 width 用来设置条形的相对宽度和条形之间的间距。其默认值是 0.8，如果将参数 width 设置为 1，则条形之间没有间距。
- bar(...,'style') 参数 style 用来设置条形的形状类型，可以选择 "group" 和 "stack"；MATLAB 的默认值是 "group"。如果选择 "stack"，则 MATLAB 会绘制累积直方图。

命令 bar3 和上面的命令大致相同，只是由于 bar3 命令绘制的是三维直方图，因此在命令 bar(...,'style')中参数style的取值可以为'detached'、'grouped',或者 'stacked'。MATLAB的默认值是'detached'。关于这些参数的具体含义可以从后面的具体实例中直观了解到。



关于命令 barh 和 bar3h，调用方法和命令 bar 和 bar3 大致相同，只是将结果的垂直条形图转换为水平条形图，可以查看相应的 MATLAB 帮助文件。

6.4.4 直方图实例

例 6.22 某公司统计了该公司 3 个部门在 5 个月内的销售情况，现在公司的销售部门需要将这些销售数据绘制成直方图，以方便公司销售部门查看和对比。在 MATLAB 中，可以根据统计的数据来绘制多种直方图。

step 1 在 MATLAB 中的命令窗口中输入下面的命令：

```
>> Income=[0.5,0.7,0.8;0.7,0.8,0.4;0.4,0.3,0.9;0.3,0.6,0.9;0.2,0.1,0.6];
>> subplot(2,2,1); bar(Income,'group');title('Group');
>> subplot(2,2,2); bar(Income,'stack');title('Stack');
>> subplot(2,2,3); barh(Income,'stack');title('Stack');
>> subplot(2,2,4); bar(Income,1.5);title('Width = 1.5');
```

step 2 查看图形结果。输入代码之后，按 "Enter" 键，得到的曲线如图 6.31 所示。

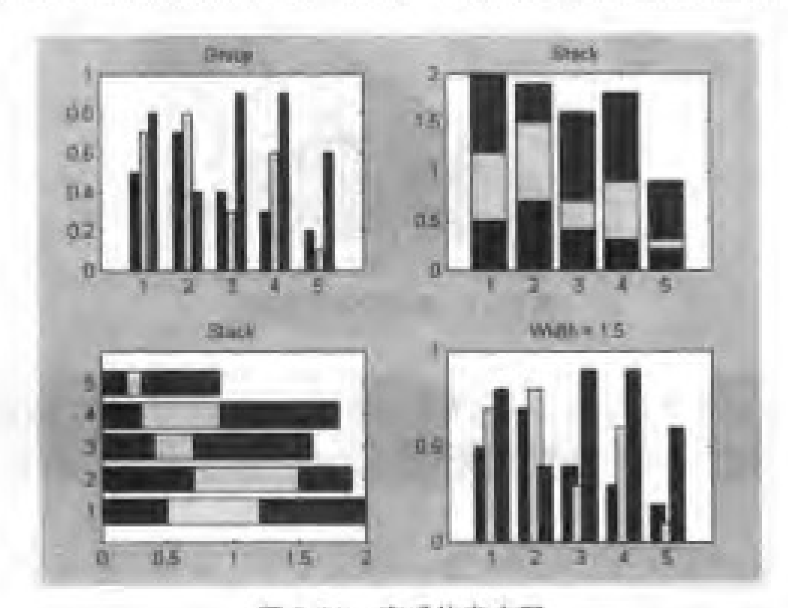


图 6.31 完成的直方图

在上面的第一个子图表中，绘制的是MATLAB中的默认直方图；第二子图绘制的是累积直方图，因为使用的参数是"stack"；第三个子图绘制的是水平累积直方图，相当于将第二个子图旋转90°得到的图形；最后一个子图是将直方图的宽度设置为1.5，从结果可以看出直方图中的各个条形有交叉的情况。

例 6.23 在MATLAB中绘制三维直方图，为了简化绘制步骤，可以使用函数cool返回的数值。

step 1 在MATLAB中的命令窗口中输入下面的命令：

```
>> Y=cool(8);
>> subplot(2,2,1);bar3(Y,'detached');title('Detached')
>> subplot(2,2,2);bar3(Y,'grouped');title('Grouped')
>> subplot(2,2,3);bar3(Y,'stacked');title('Stacked')
>> subplot(2,2,4);bar3(Y,0.3,'stacked');title('Width = 0.3');colormap
([1 0 0;0 1 0;0 0 1]);
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图6.32所示。

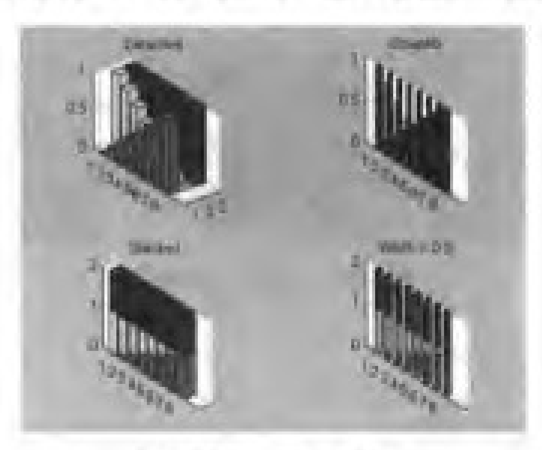


图 6.32 三维直方图

6.4.5 pie 命令

饼图是分析数据比例中常用的图表类型，主要用于显示各个项目与其总和的比例关系，它强调部分与整体的关系。在MATLAB中，提供pie和pie3命令来分别绘制二维和三维饼图。

在MATLAB中，pie的常见调用方式如下：

- pie(X) 绘制向量X的饼图，向量X中的每一个元素就是饼图中的一个扇形；
- pie(X,explode) 参数explode和向量X是同维矩阵，如果其中有非零的元素，X矩阵中对应的位置元素在饼图中对应的扇形将向外移出，加以突出；
- pie(...,labels) 参数labels用来定义对应扇形的标签。



命令pie3的参数格式和命令pie大致相同，只是最后显示的结果是一个三维的饼图，关于相关的命令请查阅相应的帮助文件。

6.4.6 二维饼图实例

例 6.24 在MATLAB中绘制二维饼图，来分析各个部门销量所占的比例。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> x = [1 2.5 1.8 3.6 2.4];
>>subplot(2,2,1);pie(x,1:5,{'Dep1','Dep2','Dep3','Dep4','Dep5'})
>>subplot(2,2,2);pie(x);colormap cool
>>subplot('position',[0.2,0.05,0.6,0.45]);explode=[1 0 0 1 0];pie(x,
explode);
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.33 所示。

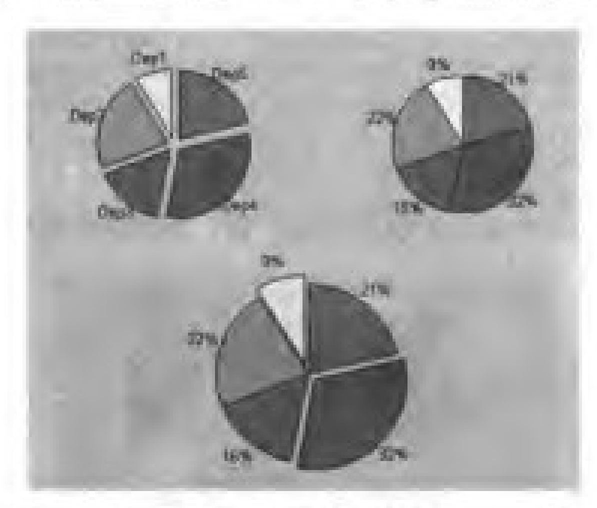


图 6.33 绘制二维饼图

6.4.7 三维饼图实例

例 6.25 在 MATLAB 中绘制三维饼图，以分析各个部门销量所占的比例。

step 1 在 MATLAB 的命令窗口中输入下面的命令：

```
>> x = [1 2.5 1.8 3.6 2.4];
>>subplot(2,2,1);pie3(x,1:5,{'Dep1','Dep2','Dep3','Dep4','Dep5'})
>>subplot(2,2,2);pie3(x);colormap cool
>>subplot('position',[0.2,0.05,0.6,0.45]);explode=[1 0 0 1 0];pie3(x,
explode);
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.34 所示。

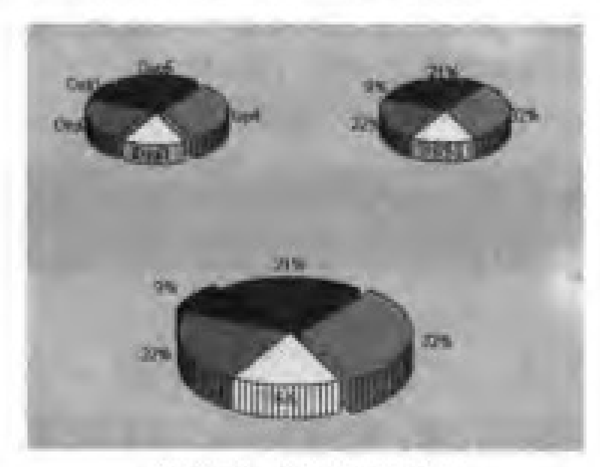


图 6.34 绘制三维饼图

6.4.8 quiver 命令

矢量图在工程领域有着广泛的应用，特别是在物理或者通信系统中有着更为重要的应用。在MATLAB 中，可以使用 quiver 命令十分便利地绘制向量场的形状。

quiver 命令的常用调用格式如下：

- quiver(x,y,u,v) 在坐标点(x,y)的地方使用箭头图形来绘制向量，其中(u,v)是对应坐标点的速度分量，参数 x，y，u，v 必须是同维向量；
- quiver(u,v) 在 x-y 平面坐标系中等距的坐标点上绘制向量(u,v)；
- quiver(...,scale) 参数 scale 是用来控制图中向量长度的实数，其默认数值是 1。可以根据需要重新设置该数值，以免绘制的向量彼此重叠。

6.4.9 矢量图实例

例 6.26 在 MATLAB 中绘制 peaks 函数得到的矢量图。

step 1 在 MATLAB 的命令窗口中输入下面的命令。

```
>> n = -2.0:.2:2.0;
>> [X,Y,Z] = peaks(n);
>> [U,V] = gradient(Z,.2);
>>quiver(X,Y,U,V)
>> hold on
>> contour(X,Y,Z,10);
```

step 2 查看图形结果。输入代码之后，按 "Enter" 键，得到的曲线如图 6.35 所示。

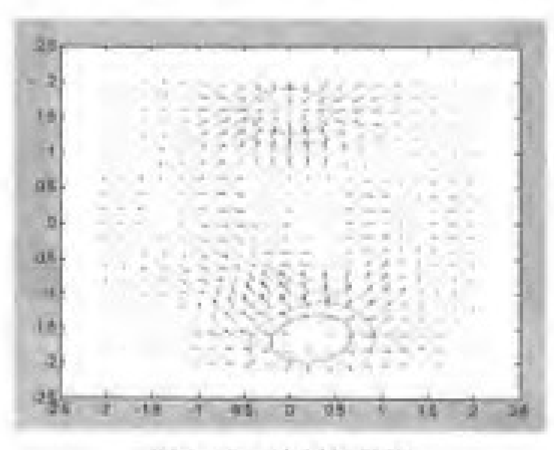


图 6.35 绘制矢量图

在上面的程序中，首先使用函数 gradient 产生 peaks 函数数据点的导数数值，然后将其结果当作 quiver 的输入数值，产生矢量图的数据；在上面程序的最后一行，使用 contour 函数绘制等高线，使整个图形更加美观。

6.4.10 contour 命令

等高线原来是属于地理领域的名词，后来等高线的概念引入数学或者物理中。在 MATLAB 中，等高线图实质上给出函数的 "地形图"，等高线把曲面上高度相同的点连在一起。在默认情况下，MATLAB 就是画出相应于一系列相等的空间 z 值的等高线。在 MATLAB 中，提供了 contour 和 contour3

命令来绘制二维和三维等高线，使用这些命令可以很方便地绘制一些比较复杂的等高线。

其中，contour 命令的常用调用方式如下：

- contour(Z) 变量 Z 就是需要绘制等高线的函数表达式；
- contour(Z,n) 参数 n 是所绘图形等高线的条数；
- contour(Z,v) 参数 v 是一个输入向量，等高线条数等于该向量的长度，而且等高线的数值等于对应向量的元素数值；
- [C,h] = contour(...) C 是等高线矩阵，h 是等高线的句柄。

6.4.11 二维等高线实例

例 6.27 在 MATLAB 中绘制 peaks 函数所得到的二维等高线。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> Z = peaks;
>> [C,h] = contour(interp2(Z,4));
>> text_handle = clabel(C,h);
>> set(text_handle,'BackgroundColor',[1 1 .6],...
    'Edgecolor',[.7 .7 .7])
>> grid on
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.36 所示。

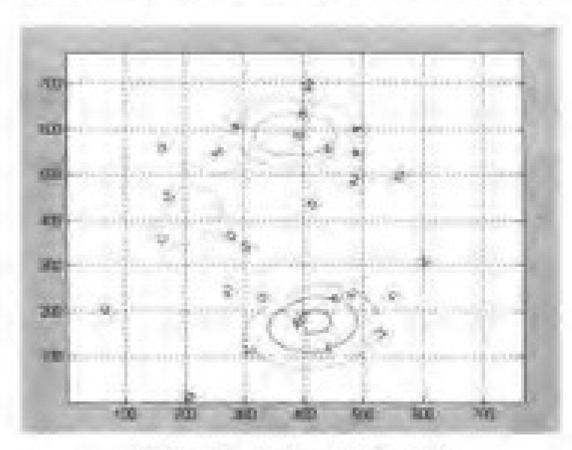


图 6.36 绘制二维等高线

6.4.12 三维等高线实例

例 6.28 在 MATLAB 中绘制函数 $z = xe^{-(x^2+y^2)}$ 的三维等高线。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> [X,Y] = meshgrid([-2:.25:2]);
>>Z = X.*exp(-X.^2-Y.^2);
>>contour3(X,Y,Z,30)
>>surface(X,Y,Z,'EdgeColor',[.8 .8 .8],'FaceColor','none')
>>grid off
>>view(-15,25)
>>colormap hot
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.37 所示。

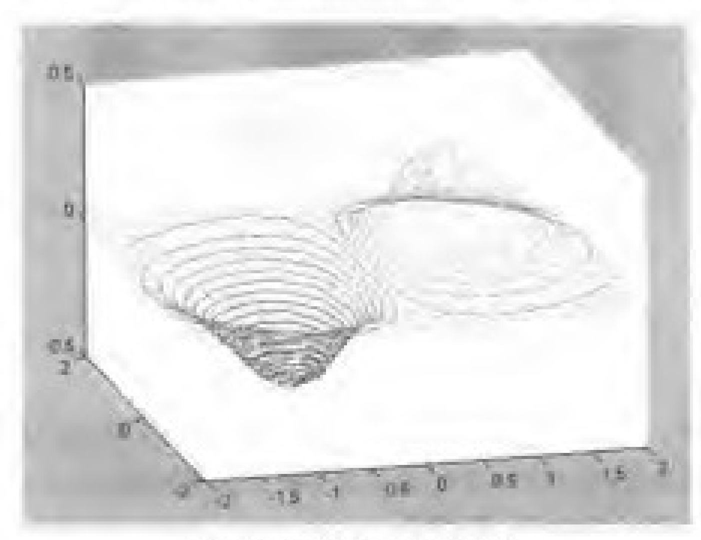


图 6.37　绘制三维等高线



在例 6.27 中，使用了 clabel 命令来为等高线设置数值，然后使用高级句柄语句来设置等高线数值的属性；在例 6.28 中，则使用了 surface 来设置三维等高线的颜色。相关函数的信息，可以查阅 MATLAB 中的帮助文件。

6.4.13 伪色彩图

在 MATLAB 中，伪色彩图（Pseudocolor）是指一个规则的矩形矩阵。该矩阵中的颜色由矩阵数值决定，对应的命令是 pcolor(C)。MATLAB 使用 C 矩阵中的每四个节点来决定一个矩形表面的颜色。伪色彩图从效果上来看，相当于从上面向下看的一个表面图。因此，命令 pcolor(X,Y,C) 相当于命令 surf (X,Y,0*Z,C) 和 view([0 90])的组合。

在 MATLAB 中，pcolor 命令的常见调用格式如下：

- pcolor(C)　绘制 C 矩阵的伪色彩图，默认情况下是线性插值绘图；
- pcolor(X,Y,C)　在平面坐标系统中的(x,y)坐标点绘制 C 矩阵的伪色彩图。

例 6.29　在 MATLAB 中绘制函数 peaks 的伪色彩图。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> [ x,y,z] =peaks(35);
>> pcolor(x,y,z);
>> colormap hsv
>> shading interp
>> hold on; C=contour(x,y,z,4,'k') ;
>> clabel(C);
>> zmax=max(max(z));zmin=min(min(z));caxis([ zmin,zmax] );
>> colorbar;
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.38 所示。

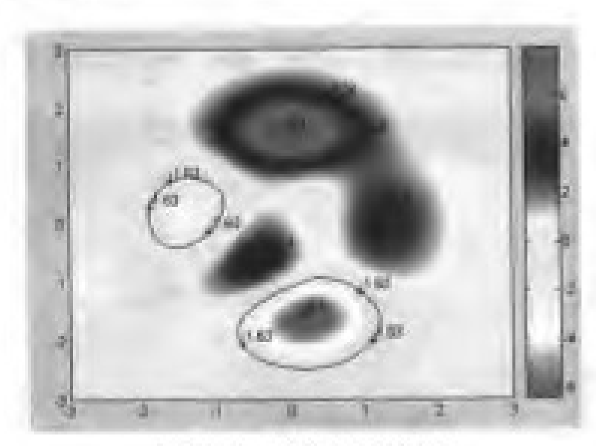


图 6.38　绘制伪色彩图

在上面的程序代码中，使用了 shading interp 来设置图形的明暗对比情况，然后使用前面介绍的 contour 命令来绘制等高线，并使用 clabel 命令为等高线设置数值标示。在程序的最后，为整个图形添加了颜色标尺，显示各个颜色对应的数值。

上面这些命令在编辑图形的颜色属性时是经常遇到的，在后面的章节中，将会对上面这些命令进行详细介绍。

6.4.14　errorbar 命令

误差线在数据分析中有着重要的应用。在原始图形中添加数据误差线，可以方便用户直观地查看各个数据点的误差变化范围。在 MATLAB 中，为用户提供了 errorbar 命令，可以绘制沿着一曲线画出误差棒形图，其中误差棒是数据的置信水平或者沿着曲线的偏差。当命令输入参数是矩阵时，则按矩阵的数据列画出误差棒。

在 MATLAB 中，errorbar 命令的常见调用格式如下：

- errorbar(Y,E)　绘制向量 Y 的曲线，然后显示在向量 Y 的每个元素的误差棒。误差棒表示曲线 Y 上面和下面的距离，因此误差棒的长度是 2E；
- errorbar(X,Y,E)　X、Y 和 E 必须是同类型的参量。如果三个参数都是向量，MATLAB 会绘制出带长度为 2E、对称误差棒于曲线点；如果三个参数都是矩阵，MATLAB 则会绘制出带长度为 E(i, j)，并对称误差棒于曲面点。

6.4.15　误差棒形图实例

例 6.30　在 MATLAB 中绘制函数 $y=\dfrac{1}{(x-3)^2+1}+\dfrac{1}{(x-9)^2+4}+5$ 的图形，同时绘制对应数据点的误差棒。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> x=0:0.5:16;
>> y= 1 ./ ((x-3).^2 + 1) + 1 ./ ((x-9).^2 + 4) + 5;
>> E = std(y)*ones(size(x));
>> errorbar(x,y,E,'r','LineWidth',2);
>> grid on
>> box on
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.39 所示。

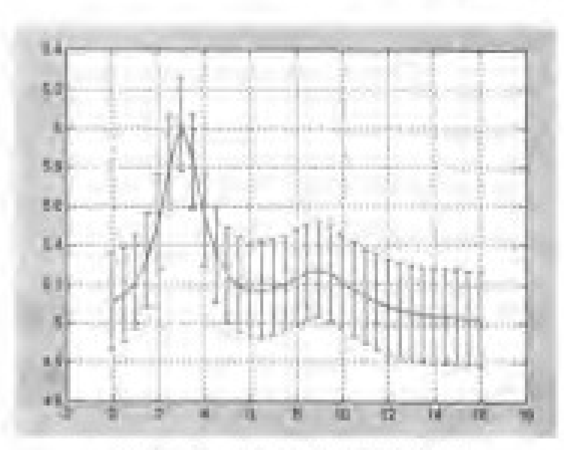


图 6.39 绘制曲线的误差线

在上面的程序中，使用程序代码E = std(y)*ones(size(x))得到每个数据点的误差范围。其中，std(y)函数计算得到函数系列y的标准差。为了能够更好地显示误差线，将自变量x的间隔设置得较大，但是为了保持曲线特性，也不能将其设置得过大。由于矩阵E中的所有元素的大小相等，因此，曲线中每点的误差棒长度相等。



从上面的操作步骤中可以看出，误差棒线的绘制图可以像普通图形那样进行编辑，例如添加分格线、设置坐标轴属性、设置绘制曲线的属性等。

6.4.16 stem 命令

离散杆图也是一种常见的图形类型。这种图形沿着x坐标轴将坐标点用直线和基准线相连，也就相当于从数据点向坐标轴作垂线，数据点由数据标记来显示。在MATLAB中，提供二维和三维离散杆图的绘图命令stem和stem3。

在MATLAB中，绘图命令stem的常见调用格式如下：

- stem(Y)　绘制向量Y的离散杆图，由于没有参数X的信息，默认情况下X是沿着x坐标轴等距的、系统自动产生的数值系列；
- stem(X,Y)　参数X和Y必须是同维向量，绘制以X为横坐标，Y为纵坐标的离散杆图；
- stem(...,LineSpec)　其他参数和前面命令相同，其中LineSpec设置直线的属性。



关于命令stem3，其参数和stem大致相同，只是绘制离散杆图的参数为三维参数，其他参数都是类似的，读者可以查看MATLAB的帮助文档。

6.4.17 二维离散杆图

例 6.31　在MATLAB中绘制函数$y = e^{-\frac{t}{3}} + \sin(1+2t)$的离散杆图，同时绘制该函数的直线图形。

step 1 在MATLAB命令窗口中输入下面的命令：

```
>> t=[0:0.5:25]';
>> y=exp(-t/3)+sin(1+2*t);
>> h = stem(t,y,'fill','--');
>>set(get(h,'BaseLine'),'LineStyle',':');
```

```
>>set(h,'MarkerFaceColor','yellow');
>>hold on
>>t2=[0:0.1:25]';
>>y2=exp(-t2/3)+sin(1+2*t2);
>>plot(t2,y2,'r','LineWidth',2);
>> box on
>> set(gca,'YGrid','on')
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.40 所示。

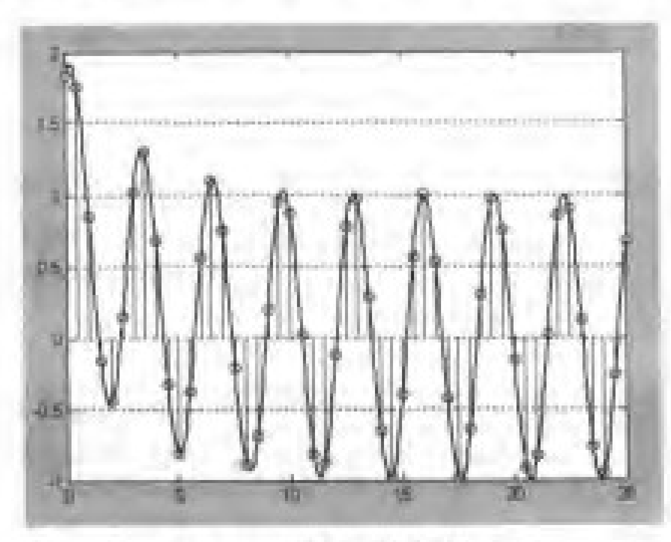


图 6.40 绘制二维离散杆图



根据上面的图形结果可以看出，离散杆图可以很直观地看出每个数据点的函数数值大小情况，离散杆图中的垂线相当于各个数据点的 Y 轴数值。

6.4.18 三维离散杆图

例 6.32 在 MATLAB 中绘制函数 $y = e^{-st}$ 的三维离散杆图。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> t = 0:.1:10;
>>s = 0.1+i;
>> y = exp(-s*t);
>> stem3(real(y),imag(y),t)
>>hold on
>>plot3(real(y),imag(y),t,'r')
>>hold off
>>view(-39.5,62)
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.41 所示。

6.4.19 散点图

在 MATLAB 中，提供 scatter、scatter3 和 plotmatrix 三种命令来绘制散点图。这三种不同的命令可以绘制不同效果的散点图。其中，scatter 命令的常见格式如下：

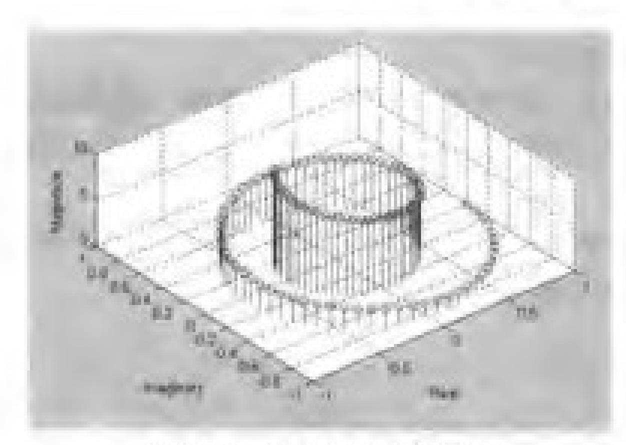


图 6.41 绘制三维离散杆图

- scatter(X,Y) 绘制关于 X 和 Y 的散点图，由于没有设置散点图的其他属性，因此 MATLAB 会采用默认的颜色和大小绘制数据点；
- scatter(X,Y,S) 参数 S 表示的是绘制数据点的颜色和大小。



命令 scatter3 和上面的命令相似，只是绘制的散点图是三维效果的，读者可以查阅 MATLAB 中的相关帮助文件。

例 6.33 在 MATLAB 中绘制柱体的三维散点图。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> [x,y,z] = cylinder(20);
>>X = [x(:)*.5 x(:)*.75 x(:)];
>>Y = [y(:)*.5 y(:)*.75 y(:)];
>>Z = [z(:)*.5 z(:)*.75 z(:)];
>>S = repmat([1.5 2.5 3.5]*25,prod(size(x)),1);
>>C = repmat([5 6 7],prod(size(x)),1);
>>scatter3(X(:),Y(:),Z(:),S(:),C(:),'filled'), view(-60,60)
```

step 2 查看图形结果。输入代码之后，按 "Enter" 键，得到的曲线如图 6.42 所示。

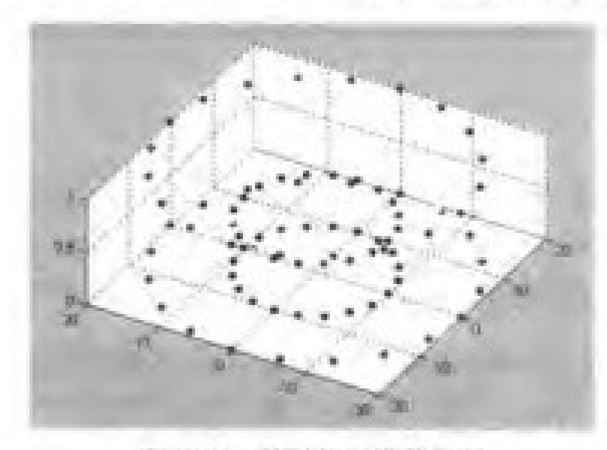


图 6.42 柱形的三维散点图



在上面的步骤中，首先使用 cylinder 命令产生三维柱体的数据，然后将其计算得到的数据赋给数组 X、Y 和 Z。在后面的程序代码中，使用 repmat、prod 函数产生散点图的颜色和大小的数组。关于这些函数的用法，可以查看 MATLAB 的帮助文件。

plotmatrix 命令的常见格式如下：

plotmatrix(X,Y,'LineSpec') 其中 X 和 Y 分别是多维矩阵，X 的维度是 $p \times n$，Y 的维度是 $p \times m$，命令 plotmatrix 将会绘制一个分割成为 $m \times n$ 个子散点图，其中第(i,j)个子散点图是根据 Y 第 i 列和 X 第 j 列数据绘制而成的。

例 6.34 在 MATLAB 中使用 plotmatrix 命令分析数据之间的统计关系。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> X=randn(150,2);
>>Y=randn(150,2);
>> subplot(1,3,2),plotmatrix(X,X)
>>subplot(1,3,1),plotmatrix(X)
>>subplot(1,3,3),plotmatrix(X,Y)
```

step 2 查看图形结果。输入代码之后，按"Enter"键，得到的曲线如图 6.43 所示。

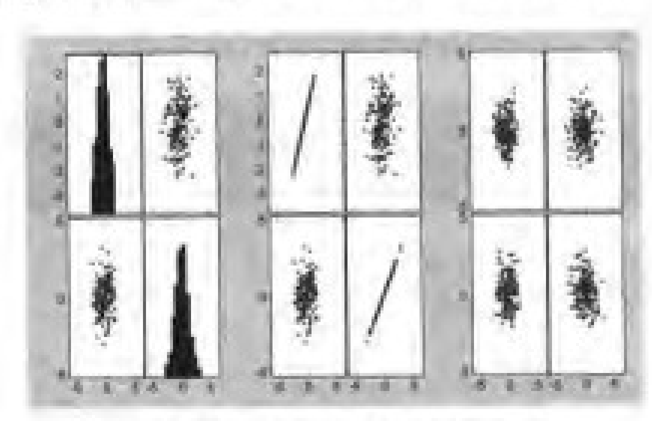


图 6.43 表现数据的统计特征



从上面的图形中可以看出，左边的子图形中的直方图表示变量 X 数据列的数据大致服从均值为 0 的正态分布，而其他散点图则表示 X 的两个数据列之间几乎是不相关的；中间的子图则表示 X 的同一列数据列是正相关的；最后的子图则表示 X、Y 之间完全不相关。

6.4.20 极坐标图形

在前面小节中，读者已经了解到 MATLAB 在直角坐标系中的绘图命令和编辑命令。在 MATLAB 中，除了可以在熟悉的直角坐标系中绘图之外，还可以在极坐标或者柱坐标中绘制各种图形。

在 MATLAB 中，绘制极坐标图形的主要命令是 polar，其常用的调用格式如下：

- polar(theta,rho) 该命令使用极角 theta 和极径 rho 绘制极坐标图形；
- polar(theta,rho,LineSpec) 参数 LineSpec 表示极坐标图形中的线条线型、标记和颜色等主要属性。

例 6.35 在 MATLAB 中绘制简单的极坐标图形。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> t = 0:.01:2*pi;
>> polar(t,sin(2*t).*cos(2*t),'--r')
```

step 2 查看图形结果。输入代码之后，按 "Enter" 键，得到的图形如图 6.44 所示。

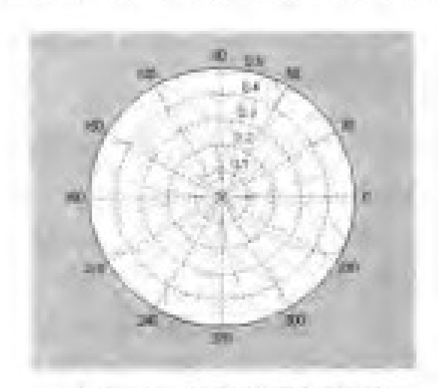


图 6.44 简易极坐标图形

6.4.21 柱坐标图形

在 MATLAB 中，绘制柱坐标图形的主要命令是 pol2cart。这个命令用于将极坐标或者柱坐标的数值转换成直角坐标系下的坐标值，然后使用三维绘图命令进行绘图，也就是在直角坐标系统下绘制使用柱坐标值描述的图形。

例 6.36 在 MATLAB 中绘制简单的柱坐标图形。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> theta=0:pi/50:4*pi; rho=sin(theta);
>> [ t,r] =meshgrid(theta,rho);
>> z=r.*t;
>> [ X,Y,Z] =pol2cart(t,r,z);
>> mesh(X,Y,Z)
```

step 2 查看图形结果。输入上面的代码后，按 "Enter" 键，得到的图形如图 6.45 所示。

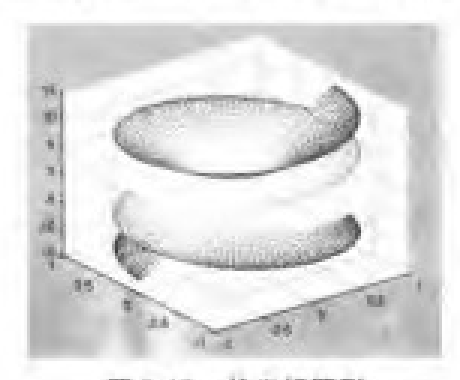


图 6.45 柱坐标图形

6.5 四维图形

对于三维图形，可以利用 z=z（x,y）的函数关系来绘制图形。该函数的自变量只有两个，从自变量的角度来讲，就是二维的。在实际生活和工程应用中，有时会遇到自变量个数为 3 的情况，这个时候自变量的定义域是整个三维空间。而计算机有显示维度，只能显示三个空间变量，不能表示第四

维的空间变量。对于这种矛盾关系，MATLAB采用了颜色、等位线等手段来表示第四维的变量。

6.5.1 slice 命令

在MATLAB中，使用slice等相关命令来显示三维函数切面图、等位线图，可以很方便地实现函数上的四维表现。slice命令的常用调用格式如下。

- slice(V,sx,sy,sz) 显示三元函数V=V(X,Y,Z)所确定的超立体形在x轴、y轴和z轴方向上的若干点的切片图，各点的坐标由数量向量sx,sy,sz来指定。
- slice(X,Y,Z,V,sx,sy,sz) 显示三元函数V=V(X,Y,Z)所确定的超立体形在x轴、y轴和z轴方向上的若干点的切片图。也就是说，如果函数V=V(X,Y,Z)有一个变量X取值X0，则函数V=V(X0,Y,Z)变成一立体曲面的切片图。各点的坐标由数量向量sx,sy,sz来指定。
- slice(V,XI,YI,ZI) 显示由参量矩阵XI,YI和ZI确定的超立体图形的切片图。参量XI,YI,ZI定义了一个曲面，同时会在曲面的点上计算超立体V的数值。
- slice(...,'method') 参数method用来指定内插值的方法。常见的方法包括有linear、cubic和nearest等，分别对应不同的插值方法。



在MATLAB中，和slice相关的命令还有contourslice和streamslice等，分别绘制出不同的切片图形。在通常的情况下，需要将上面多种切片图一起使用。

6.5.2 切片图实例

例6.37 在MATLAB中表现水体水下射流速度数据flow的切片图。

step 1 在MATLAB命令窗口中输入下面的命令：

```
>> [x y z v] = flow;
>> x1=min(min(min(x)));x2=max(max(max(x)));
>> sx=linspace(x1+1.5,x2,4);
>> slice(x,y,z,v,sx,0,0);
>> view([-33 36]);
>> shading interp;
>> colormap hsv;
>> alpha('color')
>> colorbar
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图6.46所示。

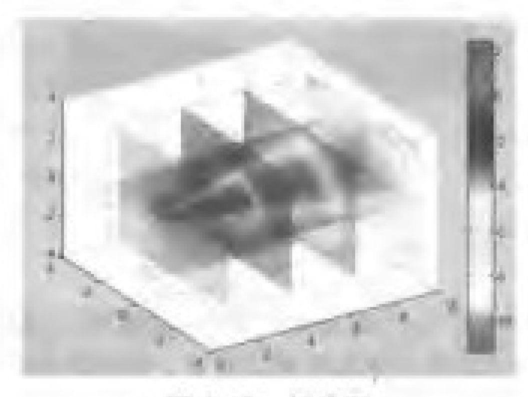


图6.46 切片图



在上面的程序代码中，最后的程序代码行都是对该三维图形进行颜色和视角的编辑工作，目的是为用户更好地查看该图形。关于这些代码的说明将在后面的章节中详细介绍。

6.5.3 切面等位线图实例

例 6.38 在 MATLAB 中表现水体水下射流速度数据 flow 的切面等位线图。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> [ x y z v] = flow;
>>x1=min(min(min(x)));x2=max(max(max(x)));
>>sx=linspace(x1+1.5,x2,4);
>>v1=min(min(min(v)));v2=max(max(max(v)));
>>cv=linspace(v1+1,v2,20);
>>contourslice(x,y,z,v,sx,0,0,cv)
>>view([ -12 30])
>>colormap cool
>>box on
>>colorbar
```

step 2 查看图形结果。输入上面的代码后，按 "Enter" 键，得到的图形如图 6.47 所示。

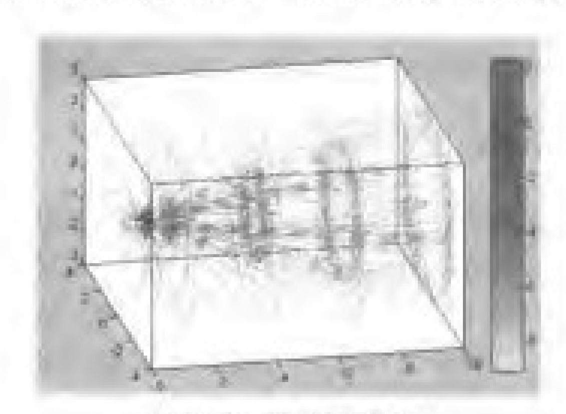


图 6.47 切面等位线图

6.5.4 流线切面图实例

例 6.39 在 MATLAB 中绘制函数 peaks 的流线切面图。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> z = peaks;
>>surf(z)
>>shading interp
>>hold on
>> [ c ch] = contour3(z,20); set(ch,'edgecolor','b')
>> [ u v] = gradient(z);
>>h = streamslice(-u,-v);
>>set(h,'color','k')
>>colormap hsv
>> for i=1:length(h);
```

```
    zi = interp2(z,get(h(i),'xdata'),get(h(i),'ydata'));
    set(h(i),'zdata',zi);
end
>>view(30,50); axis tight; colorbar
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图6.48所示。

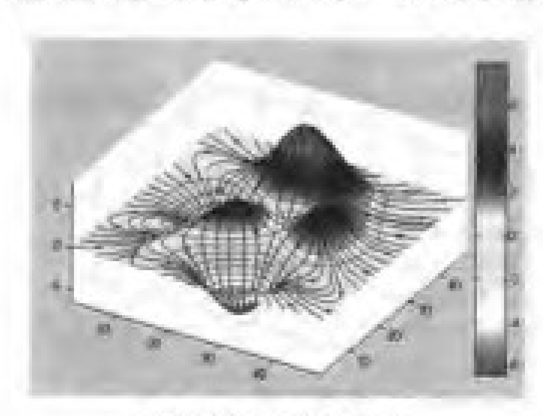


图6.48 流线切面图

6.6 三维图形的编辑

对于三维图形，除了可以像二维图形那样编辑线型、颜色外，还需要编辑三维图形的视角、材质、照明等。这些内容都是三维图形的特殊编辑工作，都是二维图形所没有的。

6.6.1 视角控制命令 view

三维视图表现一个空间内的图形，可以从不同的位置和角度来观察该图形。对于空间视图变化比较大的三维视图，使用不同的视角会产生完全不同的效果。为了对每个图形使用最佳的视觉角度，MATLAB提供对图形进行视觉控制的命令，主要有两类命令。其中，第一类命令为view，可以使用该命令来改变图形的观察点；另一类命令为rotate，使用这个命令，用户可以直接旋转三维图形。

在MATLAB中，view命令的常见调用格式如下。

- view(az,el)、view([az,el]) 该命令的功能是为三维空间图形设置观察点的方位角。方位角az(azimuth)和仰角el(elevation)是按照下面的方法定义的两个旋转角度：做一个通过用户视点和z轴的平面，该平面会和xy平面有一个交线，该交线和y轴的负方向之间有一个夹角，该夹角就是视点的方位角；在通过视点和z轴的平面上，用一条直线来连接视点和坐标原点，该直线和xy平面的夹角就是观察点的仰角。
- view([x,y,z]) 在笛卡儿坐标系中将视角设置为沿着向量[x,y,z]指向原点，例如view([0,0,1])=view(0,90)。也就是在笛卡儿坐标系中将点(x,y,z)设置为视点。
- view(2) 设置默认的二维形式视点，其中az=0，el=90。
- view(3) 设置默认的三维形式视点，其中az=-37.5，el=30。

6.6.2 view 命令实例

例6.40 在MATLAB中从不同的视角查看三维函数 $\begin{cases} z=10\dfrac{\sin 2t}{t} \\ y=0 \end{cases}$ 的图形。

step 1 在 MATLAB 命令窗口中输入下面的命令。

```
>> t=0.01*pi:0.1*pi:3*pi;
>>z=10*sin(2*t)./t;
>>y=zeros(size(t));
>>subplot(2,2,1);plot3(t,y,z,'r','LineWidth',2);grid on;title('Default
view')
>>subplot(2,2,2);plot3(t,y,z,'r','LineWidth',2);grid on;
>>title('az Rotated to 52.5');view(-37.5+90,30)
>>subplot(2,2,3);plot3(t,y,z,'r','LineWidth',2);grid on;title('El Rotated
to 10');view(-37.5,10)
>>subplot(2,2,4);plot3(t,y,z,'r','LineWidth',2);grid on;title('az=0
El=90');view(0,90)
```

step 2 查看图形结果。输入上面的代码后，按“Enter”键，得到的图形如图 6.49 所示。

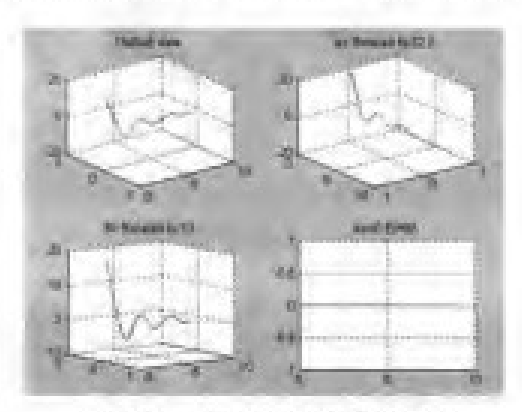


图 6.49 为三维图形设置视角



在 MATLAB 中为用户设置视点转换矩阵命令 viewmtx。该命令计算一个 4 × 4 阶的正交的或者透视的转换矩阵，该矩阵将一个四维的、齐次的向量转换到一个二维的平面中。关于该命令的信息，可以查看相应的帮助文件。

6.6.3 旋转控制命令 rotate

下面开始讲解常见的视角命令 rotate。在 MATLAB 中，rotate 命令的常见调用格式如下：

- rotate(h,direction,alpha) 该命令将图形句柄 h 的对象绕方向旋转一个角度。其中参数 h 表示的是被旋转的对象（例如线、面等）；参数 direction 有两种设置方法：球坐标设置法，将其设置为[theta,phi]，其单位是“°”（度）；直角坐标法，与就是[x,y,z]；参数 alpha 是绕方向按照右手法则旋转的角度。



和前面的命令 view 不同，如果使用命令 rotate，则通过旋转变换改变原来图形对象的数据。而命令 view 则没有改变原始数据，只是改变视角。

6.6.4 rotate 命令实例

例 6.41 在 MATLAB 中从不同的视角中查看函数 Peaks 的三维图形。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> z=peaks(25);
>>subplot(1,2,1);surf(z);title('Default')
>>subplot(1,2,2);h=surf(z);title('Rotated')
>>rotate(h,[-2,-2,0],30,[2,2,0])
>>colormap cool
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 6.50 所示。

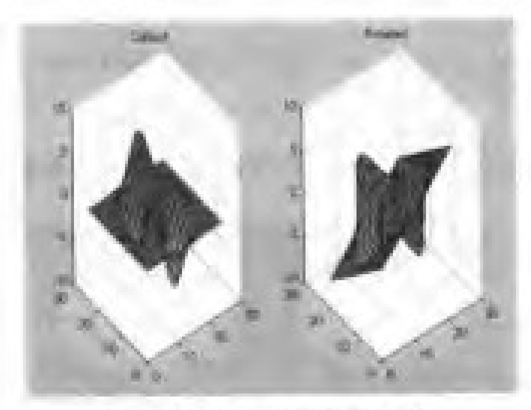


图 6.50 图形对象的旋转



从上面两幅图形可以看出，使用命令 view 旋转的是坐标轴，而使用命令 rotate 旋转的是图形对象本身，而坐标轴保持不变。

在 MATLAB 中，还提供一个动态旋转命令 rotate3d。使用该命令可以动态调整图形的视角，直到用户觉得合适为止，而不用自行输入视角的角度参数。下面使用一个简单的实例来说明如何使用命令 rotate3d。

例 6.42 在 MATLAB 中动态调整函数 Peaks 的三维图形的视角。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> surf(peaks(40));
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 6.51 所示。

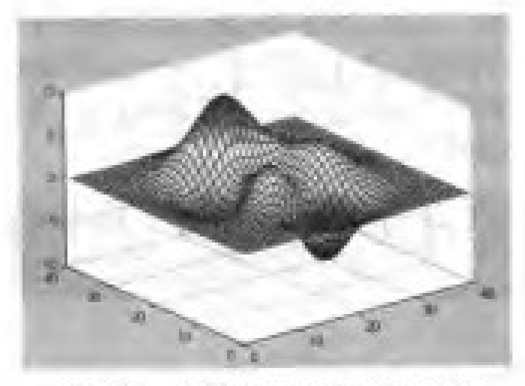


图 6.51 原始的 pesks 函数图形

step 3 在 MATLAB 命令窗口中输入下面的命令：

```
>> rotate3d;
```

step 4 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 6.52 所示。

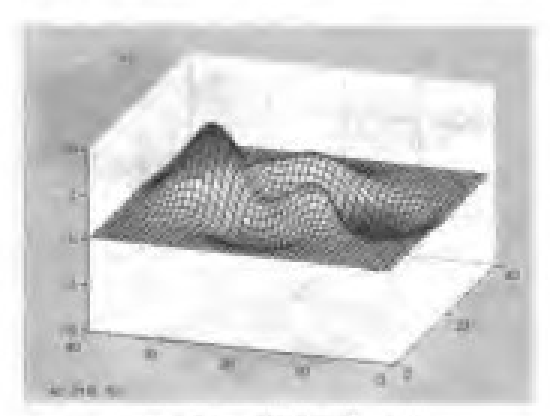


图 6.52 旋转三维图形

从上面的图形中可以看出，当输入 rotate3d 命令后，图形中出现旋转的光标。可以在图形窗口的区域中按住鼠标左键来调节视角，在图形窗口的左下方出现所调整的角度，在本例中是 Az=-71，El=16。



为了能够在MATLAB中获得最好的视觉效果，可以先使用rotate3d命令让鼠标来调节视角，再使用指令将调整后的视角加以固定。例如在本例中，首先使用 rotate3d 命令将图形调整到合适的视角，然后将图形中显示的视角数值设置到 view 函数中。

6.6.5 设置背景颜色

图形的色彩是图形的主要表现因素，丰富的颜色变化可以让图形更具有表现力。在 MATLAB 中，提供多种色彩控制命令，这些命令分别适用于不同的环境，可以对整个图形中的所有因素进行颜色设置。

在 MATLAB 中，设置图形背景颜色的命令是 colordef，该命令的常用调用格式如下：

- colordef white 将图形的背景颜色设置为白色；
- colordef black 将图形的背景颜色设置为黑色；
- colordef none 将图形背景和图形窗口的颜色设置为默认的颜色；
- colordef(fig,color_option) 将图形句柄fig图形的背景设置为由color_option设置的颜色。



colordef 命令属于设置图形对象的上层命令，它将影响其后产生的图形窗口中的所有图形对象的颜色设置。当在第一个参数fig中指定设置颜色的图形窗口的图柄或者字符串后，其影响就限于指定的图形窗。

例 6.43 在 MATLAB 中为函数 peaks 的图形设置不同的背景颜色。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> subplot(2,2,1);colordef none;surf(peaks(35));Title('Default');
>> subplot(2,2,2);colordef black;surf(peaks(35));Title('Black');
>> subplot('position',[ 0.2,0.05,0.6,0.45]);
>> colordef white;surf(peaks(35));Title('White')
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图6.53所示。

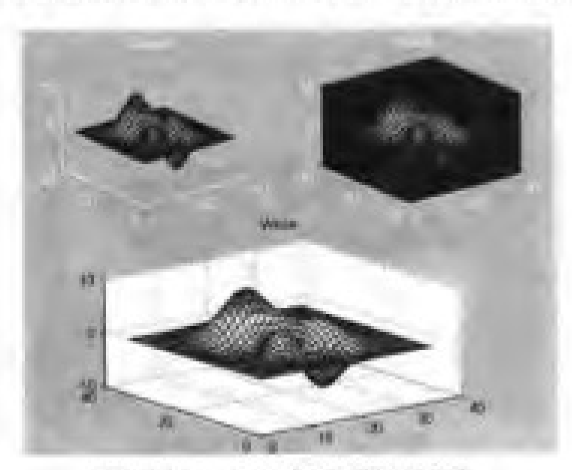


图6.53 设置不同的背景颜色



说明：在MATLAB中，除了可以设置图形的背景颜色之外，还可以设置图形中坐标轴的颜色，其相关的命令为whitebg。限于篇幅，本书中就不展开分析该命令，感兴趣的读者可以查阅MATLAB的帮助文件。这里只是提示读者，whitebg命令只影响当前或者指定图形窗口的坐标轴的颜色。

6.6.6 设置图形颜色

从上面小节的内容中可以看出，在MATLAB中可以很方便地设置图形的背景颜色；如果需要修饰图形的颜色，则还需要使用其他的命令。在MATLAB中处理图形颜色的重要命令是colormap。在前面小节中，读者已经多次接触到该命令，在本小节中，将详细介绍该命令的使用方法以及相应的原理。

MATLAB采用颜色映像来处理图形颜色，也就是RGB色系。在MATLAB中，每种颜色都是由三个基色的数组表示的。数组元素R、G和B在[0,1]区间取值，分别表示颜色中红、绿、蓝三种基色的相对亮度。通过对R、G、B大小的设置，可以调制出不同的颜色。在MATLAB中，当使用绘图命令时，所有线条的颜色都是通过RGB调制出来的，表6.4列出一些常见的颜色配比方案。

表6.4 常见的颜色配比方案

基色			调制的颜色	对应的MALTAB符号
R	G	B		
0	0	1	蓝色（Blue）	b
0	1	0	绿色（Green）	g
1	0	0	红色（Red）	r
0	1	1	青色（Cyan）	c
1	0	1	品红色（Magenta）	m
1	1	0	黄色（Yellow）	y
0	0	0	黑色（Black）	k
1	1	1	白色（White）	w

当调好相应的颜色后，就可以使用MATLAB中的常见绘图命令来调用这些颜色，例如mesh、surf等。调用色图的基本命令是：

```
colormap([ R,G,B] )
```

其中，函数的变量[R,G,B]是一个三列矩阵，行数不限，这个矩阵就是所谓的色图矩阵。在MATLAB中，每一个图形只能有一个色图。色图可以通过矩阵元素的直接赋值来定义，也可按照某个数据规律产生。

MATLAB预定义了一些色图矩阵CM数值，它们的维度由其调用格式来决定：

- CM 返回维度为 64 × 3 的色图矩阵；
- CM(m) 返回维度是 m × 3 的色图矩阵。

表6.5列出了MATLAB中的色图矩阵名称以及含义。

表6.5 MATLAB中的CM名称

名称	含义	名称	含义
autumn	红、黄色图	bone	蓝色调灰度色图
cool	青、品红浓淡色图	copper	纯铜色调浓淡色图
gray	灰色调浓淡色图	hot	黑红黄白色图
hsv	饱和色图	jet	蓝头红尾的饱和色图

例6.44 在MATLAB中绘制函数peaks的图形，同时设置该图形的颜色。

step 1 在MATLAB命令窗口中输入下面的命令：

```
>> surf(peaks(100));
>> colormap(cool(512))
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图6.54所示。

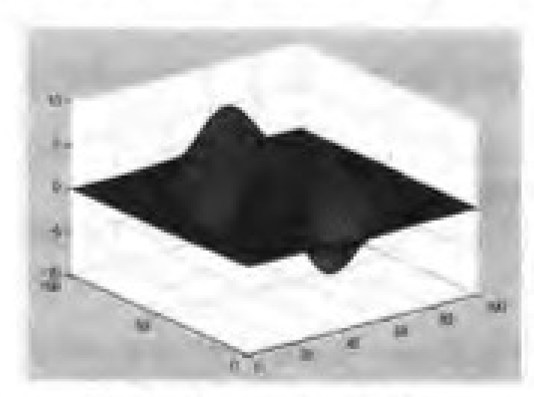


图6.54 使用cool绘图

6.6.7 设置数值轴的颜色

除了colormap函数外，MATLAB还提供多种颜色设置命令，设置图形中其他元素的颜色特性。其中，caxis命令和colorbar命令是经常使用的命令，下面将详细介绍这两个命令的调用方式。

在MATLAB中，caxis命令的主要功能是设置数值轴的颜色。控制数值和色彩间的对应关系的常用调用格式如下：

- caxis([cmin cmax]) 在[cmin cmax]范围内与色图矩阵中的色值相对应，并依此为图形着色。

如果数据点的数值小于 cmin 或大于 cmax，则按照等于 cmin 或 cmax 来进行着色；

- caxis auto　MATLAB 自动计算出色值的范围；
- caxis manual　按照当前的色值范围来设置色图范围；
- caxis(caxis)　和 caxis manual 实现相同的功能。

例 6.45　在 MATLAB 中绘制函数 peaks 的图形，同时设置该图形的颜色。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> z=peaks(45);
>> surf(z);
>> caxis([-1.5 1.5])
```

step 2 查看图形结果。输入上面的代码后，按 "Enter" 键，得到的图形如图 6.55 所示。

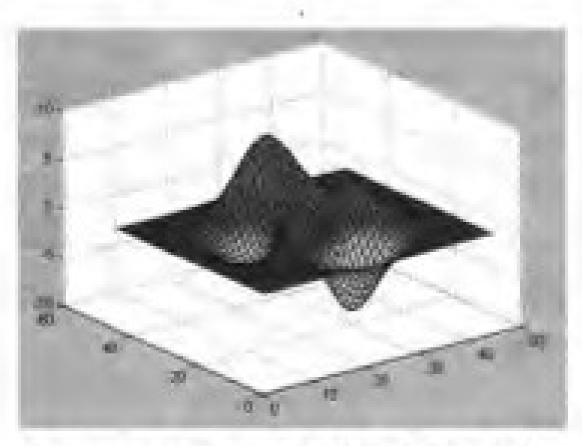


图 6.55　使用 caxis 命令为图形设置颜色

6.6.8 添加颜色标尺

在 MATLAB 中，colorbar 命令的主要功能是显示指定颜色刻度的颜色标尺，其常用调用格式如下：

- colorbar　更新最近生成的颜色标尺；如果当前坐标轴系统中没有任何颜色标尺，则在图形的右侧显示一个垂直的颜色标尺；
- colorbar('vert')　添加一个垂直的颜色标尺到当前的坐标轴系统中；
- colorbar('horiz')　添加一个水平的颜色标尺到当前的坐标轴系统中。

例 6.46　在 MATLAB 中绘制函数 peaks 的图形，同时在图形中添加水平颜色标尺。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> z=peaks(45);
>> surf(z);
>> caxis([-1.5 1.5])
>> colorbar('horiz')
```

step 2 查看图形结果。输入上面的代码后，按 "Enter" 键，得到的图形如图 6.56 所示。

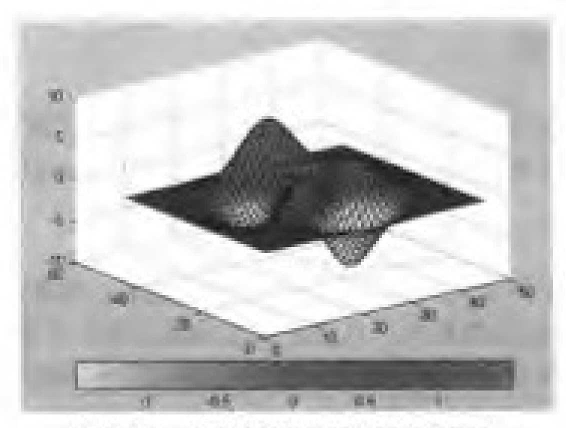


图 6.56　为图形添加水平颜色标尺



在上面的图形中，只是向前面步骤完成的图形中添加水平颜色标尺，但是其数值信息却更加详细。可以通过对比标尺中的数值和图形中的颜色，查看对应数据点的数值。

例 6.47　在MATLAB中绘制函数peaks的图形，使用不同的色图矩阵来为该图形设置不同的颜色。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>>z=peaks(30);
>>zmin=min(min(z));
>>zmax=max(max(z));
>>dz=zmax-zmin;
>>cm=[ hot;winter];
>>subplot(2,2,1);surf(z);
>>caxis([ zmin+dz*0.4,zmax])
>>colorbar('horiz')
>>subplot(2,2,2);surf(z);colorbar('horiz')
>> subplot('position',[ 0.2,0.05,0.6,0.45]);surf(z);
caxis([ zmin,zmax+dz*0.8]);colorbar('vert')
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 6.57 所示。

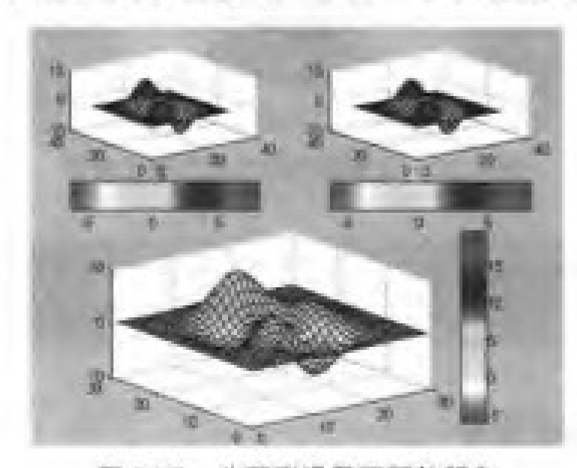


图 6.57　为图形设置不同的颜色



从上面的图形中可以看出，当为图形设置不同的颜色矩阵时，同一个图形会显示不同的颜色，同时配有颜色标尺可以很清楚地查看对应的数值。

6.6.9 设置图形的着色

在MATLAB中，除了可以为图形设置不同的颜色之外，还可以设置颜色的着色方式。对于绘图命令mesh、surf、pcolor、fill等创建的图形非数据处的着色，由shading命令决定。在MATLAB中，shading命令有三种参数选项。

- shading flat 使用平滑方式着色。网格图的某条线段，或者曲面图中的某整个贴片都是一种颜色，该颜色取自线段的两端，或者该贴片四顶点中下标最小那点的颜色。
- shading interp 使用插值的方式为图形着色。使用网格图线段，或者曲面图贴片上各点的颜色由该线段两端，或者该贴片四顶点的颜色线性插值而得。
- shading faceted 以平面为单位进行着色，是系统默认的着色方式。在flat用色基础上，再在贴片的四周勾画黑色网线。

例 6.48 在MATLAB中绘制圆柱图形，然后使用三种不同的着色方式为图形着色。

step 1 在MATLAB命令窗口中输入下面的命令：

```
>> t=0:pi/5:4*pi; [x,y,z]=cylinder(2+sin(t));
>> subplot(2,2,1);surf(x,y,z);shading interp;Title('interp')
>> subplot(2,2,2);surf(x,y,z);shading flat;Title('flat')
>> subplot('position',[0.2,0.05,0.6,0.45]);surf(x,y,z);Title('faceted');
colormap(hsv)
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图6.58所示。

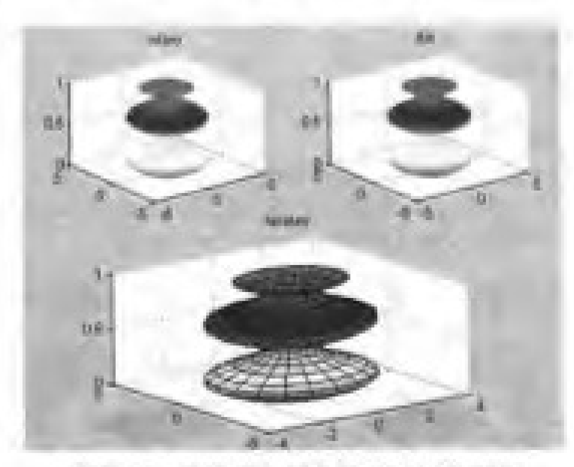


图6.58 设置图形的不同的着色方式



实质上，命令shading是编辑图形的高层命令，相当于设置当前轴上的"面"对象的EdgeColor和FaceColor属性；可以使用后面章节中的底层命令得到相同的效果。

6.6.10 照明控制 light 命令

在三维图形中，除了填充颜色和着色处理之外，还需要设置图形的灯光设置、照明模式和反射光处理，这样的图形才能显得更加美观。在本小节中，将介绍关于三维图形照明控制的相关命令，灵活使用这些命令可以使三维图形显得更加真实。

首先需要介绍灯光light命令，这是MATLAB中进行光照的基础命令，功能是为图形建立光源，其调用格式如下：

```
light('PropertyName',PropertyValue,...)
```

其中，PropertyName属性名是一些光源的颜色、位置和类型等的变量名，更为详细的介绍可以查阅相应的帮助文件。

关于light命令，需要了解的信息如下：

- 在使用这个命令之前，MATLAB会对图形采用强度各处相等的漫射光源。如果开始使用该命令，光源本身并不会出现，但是图形中的各个对象所有和“光”相关的属性都会被激活；
- 该命令的参数可以省略，当用户不输入任何参数的时候，MATLAB采用默认设置的光照：白光、无限远、透过[1，0，1]射向坐标轴；
- 对于该命令中的位置属性，可以选择两个数值：infinite和local。前者表示光照的位置是无限远，后者表示光照的位置是近光。

6.6.11 light 命令实例

例 6.49 在MATLAB中绘制peaks函数的三维图形，然后使用不同的照明效果。

step 1 在MATLAB命令窗口中输入下面的命令：

```
>> subplot(2,1,1);surf(peaks);light;title('Default')
>>subplot(2,1,2);surf(peaks);light('color','r','Position',[ 0 1 0],
'Style','local');
>> title('Red-Local Light')
```

step 2 查看图形结果。输入上面的代码后，按“Enter”键，得到的图形如图6.59所示。

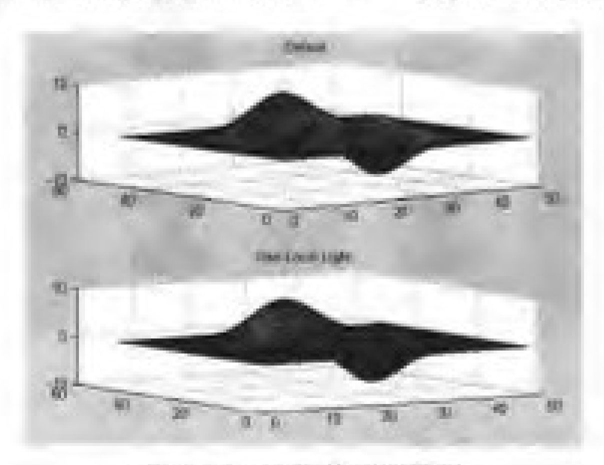


图 6.59 不同的照明控制



在图6.59中，上面的“Default”子图形采用的是默认光照效果，下面的“Red-Local Light”子图形采用的光源是红色的近光，而且射向图形的角度是[1 0 1]。从上面的图形中可以明显地查看出光照对图形显示的影响。

6.6.12 照明控制 lighting 命令

除了和灯光相关的light命令外，MATLAB还提供设置曲面光源模式的lighting命令。使用该命

令可以显示不同的照明模式，但是lighting命令必须在light命令执行后才能起作用。该命令的调用格式如下。

- lighting flat　平面模式，这是系统的默认模式。入射光均匀洒落在图形对象的每个面上，主要和faced一起配合使用。
- lighting gouraud　点模式。先对顶点颜色进行插补，再对顶点勾画的面色进行插补。
- lighting phong　对顶点处法线插值，再计算像素的反光。其表现效果最好，但是比较耗时。
- lighting none　关闭所有的光源。

6.6.13 lighting 命令实例

例 6.50　在MATLAB中绘制圆柱图形的三维图形，然后使用不同的照明效果。

step 1 在MATLAB命令窗口中输入下面的命令：

```
>> t=0:pi/20:2*pi;
>> [x,y,z]=cylinder(2+cos(t));
>>subplot(2,2,1);mesh(x,y,z);light;lighting phong;Title('Phong')
>>subplot(2,2,2);surf(x,y,z);light;shading faceted;lighting flat;Title
('Flat')
>>subplot(2,2,3);surf(x,y,z);light;shading interp;lighting gouraud;Title
('Gouraud')
>>subplot(2,2,4);surf(x,y,z);light;lighting none;Title('None')
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图6.60所示。

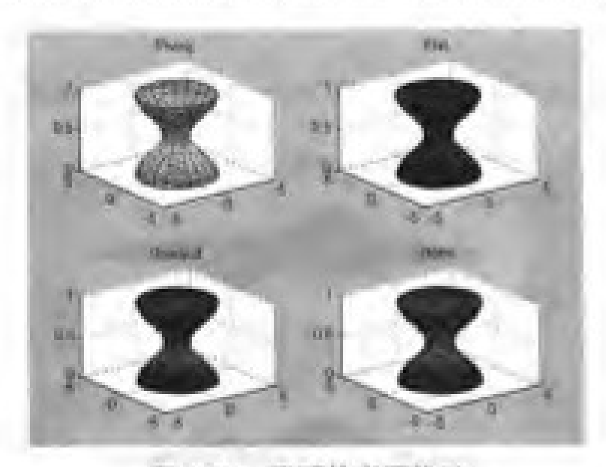


图 6.60　不同的光照效果

6.6.14 材质控制 material 命令

material命令可以控制光照效果的材质属性，也就是设置图形表面对光照反射的模式，其常用的调用格式如下。

- material options　该命令使用预定义的反射模式，对于options的不同选项，其对应的选项含义如下：
  - shiny　使对象比较明亮，镜反射份额较大，反射光的颜色取决于光源颜色；
  - dull　使对象比较暗淡，漫反射份额较大，反射光的颜色取决于光源颜色；

- metal 使对象有金属光泽，反射光的颜色取决于光源颜色和图形表面的颜色，这是MATLAB内部的默认设置；
- default 返回到MATLAB中的默认设置。

✦ material([ka kd ks n sc]) 该命令可以使用专门的个性化的设置，可以对反射的要素进行直接的设置，对应参数的含义如下：

- ka 设置无方向性、均匀的背景光的强度；
- kd 设置无方向性的漫反射的强度；
- ks 设置有硬反射光的强度；
- n 设置控制镜面亮点大小的镜面指数；
- sc 设置镜面颜色的反射系数。

6.6.15 material 命令实例

例 6.51 在 MATLAB 中绘制 peaks 函数的三维图形，同时设置不同的光照效果。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> [x,y,z]=peaks(25);
>>subplot(1,2,1);surf(x,y,z);shading interp;
>>material([0.5,0.3,0.5,10,0.5])
>>light('color','r','Position',[0 1 0],'Style','local')
>>lighting phong
>>subplot(1,2,2);surf(x,y,z);shading flat;
>>material shiny;
>>light('color','w','Position',[-1 0.5 1],'Style','local')
>>lighting flat
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 6.61 所示。

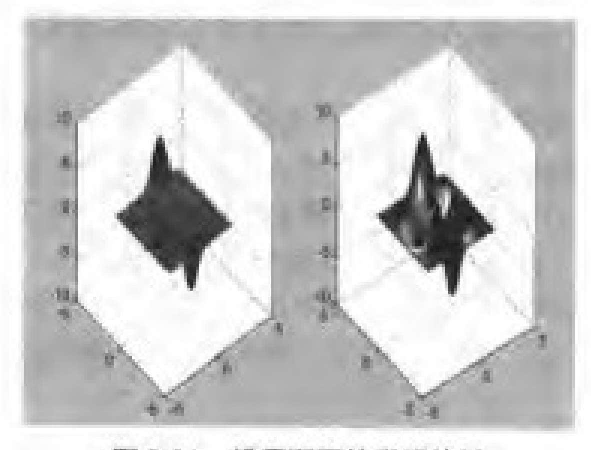


图 6.61 设置不同的光照效果

6.6.16 透视控制

在 MATLAB 中，如果使用 mesh、surf 等命令绘制三维图形，在默认情况下，MATLAB 会隐藏重叠在后面的网格线；有时需要了解隐藏的网格线，这个时候用户需要使用透视控制命令。在 MATLAB 中，透视控制命令如下：

- hidden off　透视被缩压的图形
- hidden on　消隐被缩压的图形

例 6.52　在 MATLAB 中演示透视效果。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> [x,y,z]=ellipsoid(0,0,0,1.2,2.5,4.5); [x0,y0,z0]=sphere(40);
>>surf(x0,y0,z0);shading interp
>>hold on,mesh(x,y,z),colormap(hsv),hold off
>>hidden off
>> axis equal
>>axis off
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 6.62 所示。

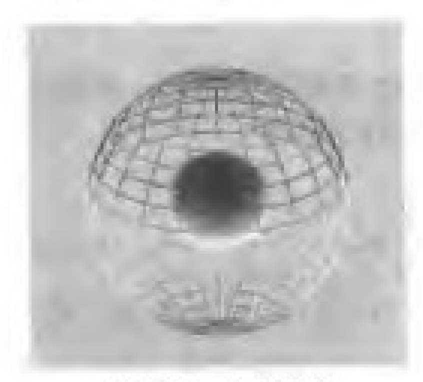


图 6.62　透视球体

6.6.17 透明控制原理

从 MATLAB 6.x 版本开始，系统增加了透明（Transparency）处理的相关命令。该命令的主要功能是使用透明技术来显示复杂图形的内部结构，和前面章节介绍的色彩和光照控制类似，透明技术也可以为数据显示提供可视化的手段。

在 MATLAB 中，将透明度设置为[0，1]区间上的数值，其中 0 表示图形是全透明的，1 表示图形是不透明的，这种量化的数值被称为 Alpha 值。在 MATLAB 中，每个图形窗口中都有一个透明表。在默认情况下，图形透明表是一个数组，其元素的数值都在[0，1]区间中取值，其中第一个元素数值为 0，最后一个元素的数值为 1，其他元素按照均匀递增方式进行排列。

限于篇幅，本书对于图形透明设置的原理就不详细介绍了。在本小节中，主要介绍 MATLAB 中透明度的处理方式。以根据 3 个 $m \times n$ 数值矩阵 X、Y 和 Z 所绘制得到的曲面为例，MATLAB 有如下三种透明度的处理方式。

- **标量**：使所有的数据点都设置相同的透明度。
- **线性数据**：使曲面数据点的透明度按照某个指定维度的方向线性变化。
- **矩阵**：使每个数据点选取不同的透明度。

上面的处理方式分别对应 MATLAB 中的 Alpha 函数中的参数，当参数是标量时，曲面中的所有数据点都是相同的透明度；当 Alpha 函数中的参数是线性数据时，曲面的透明度按照某个维度的方向线性变化。

除了使用Alpha函数分别设置曲面数据点的透明度之外，MATLAB还提供Alim函数来设置透明度的上下限，将其上下限设置为$[A_{min}, A_{max}]$。在MATLAB中设置透明度的上下限需要和上下限模式配合使用。MATLAB的对应命令是Alim，其参数就是对应设置Alpha轴的上下限。

6.6.18 透明控制实例

例 6.53 在MATLAB中显示Peaks函数的不同线性透明度效果。

step 1 在MATLAB命令窗口中输入下面的命令：

```
>> [x,y,z]=peaks(45);
>>subplot(2,1,1);surf(x,y,z);
>>shading interp
>>alpha(x);title('Along X')
>>subplot(2,1,2);surf(x,y,z);
>>shading interp
>>alpha(y);title('Along Y')
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图6.63所示。

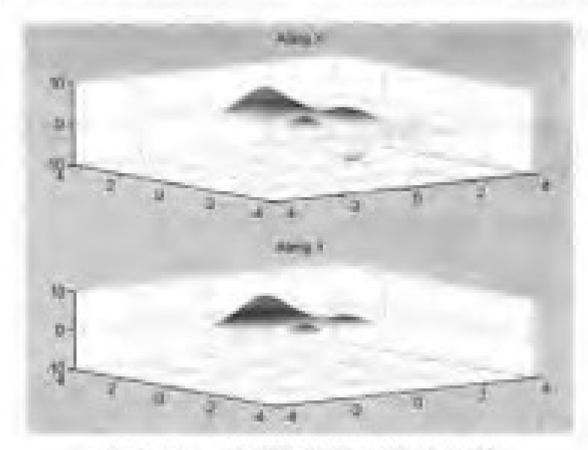


图 6.63 设置不同的线性透明度



从上面的图形结果中可以看出，上面的一幅图透明度沿着X轴不断加强，下面的一幅图的透明度则沿着Y轴不断加强。

例 6.54 在MATLAB中显示Peaks函数的三维图形，将图形的上半部分设置为不透明，下半部分设置为全透明。

step 1 在MATLAB命令窗口中输入下面的命令：

```
>> [x,y,z]=peaks(45);
>> surf(x,y,z);shading interp;
>> alpha(z)
>> Amin=-3;Amax=3;
>> alim([Amin,Amax])
>> alpha('scaled')
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图6.64所示。

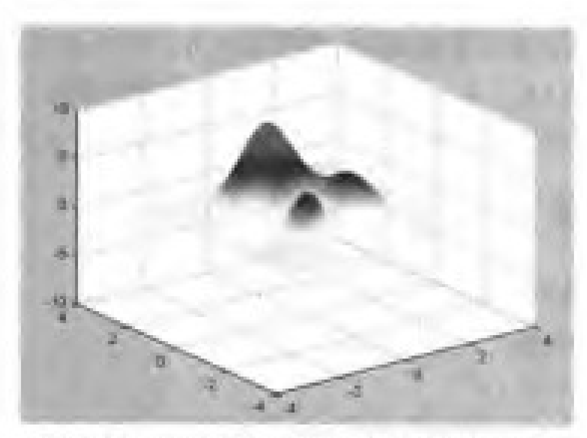


图 6.64　设置图形上下两个部分的透明度



在 MATLAB 中，当函数 alpha 中的参数是数值矩阵或者标量时，其得到的结果就是不同的透明度设置情况。alpha 函数还可以选择其他的参数类型，设置图形透明度映射方法，参数分别是 none、scaled 和 direct，对应不同的透明度映射方式。在本例中，选择 scaled 的映射方式表示将 Alim 上下限映射到透明表的一段。其他映射方式可以参考 MATLAB 的帮助文件。

例 6.55　在 MATLAB 中显示 Peaks 函数的三维图形，将图形的透明度设置为 V 型，也就是在 z 方向上中部最透明，上下两部分则最不透明。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> [x,y,z]=peaks(50);
>> surf(x,y,z);
>> shading interp
>> alpha(z)
>> alpha('interp')
>> alphamap('vdown')
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 6.65 所示。

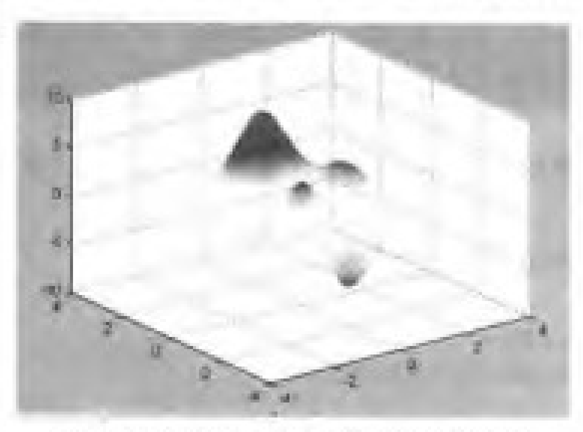


图 6.65　设置三维图形的特殊透明度



在上面的图形中，使用 alphamap 命令来设置图形窗的透明度表，参数数值为 vdown，表示图形的透明度情况是中部最透明，上下两部分则最不透明。关于 alphamap 命令的详细信息，可以参考 MATLAB 的相关帮助文件。

6.7 三维图形的简易命令

和绘制二维函数的图形类似，在MATLAB中，绘制三维函数的图形，同样也有一些简易命令。和三维绘图常见的各种命令相对应，三维图形的简易命令包括有ezmesh、ezmeshc、ezsurf和ezsurfc等。

这些命令都是在对应的绘图命令前面添加了ez的字样，表示一种简易命令。关于这些常见命令的调用格式和前面小节中简易命令相似，就不重复介绍了。本节将使用各种实例来说明这些函数的用法。

例6.56 在MATLAB中，使用三维图形的简易命令来绘制函数$f(x,y)=\frac{y}{1+x^2+y^2}$的曲面线及等高线。

step 1 在MATLAB命令窗口中输入下面的命令：

```
>>ezmeshc('y/(1 + x^2 + y^2)',[-5,5,-2*pi,2*pi])
>>colormap jet
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图6.66所示。

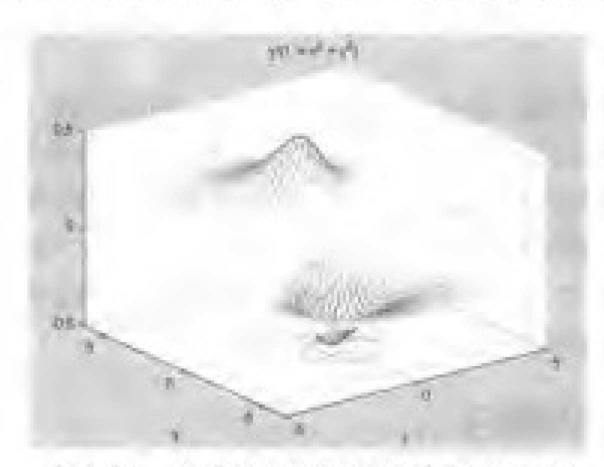


图6.66 绘制三维图形网格线以及等高线



和前面介绍的二维简易命令类似，只是指定了二元函数中参数的定义域，和整个三维函数的表达式，通过简易命令就可以绘制出响应的图形。

例6.57 在MATLAB中，在圆域上绘制函数$f(x,y)=x^2+y^2$的图形，同时为该函数图形设置相应的颜色和光照信息。

step 1 在MATLAB命令窗口中输入下面的命令：

```
>>ezsurf('x^2+y^2','circ')
>> shading flat
>> light('color','y','Position',[1 0 0],'Style','local')
>> colormap jet
>> view([-18,28])
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图6.67所示。

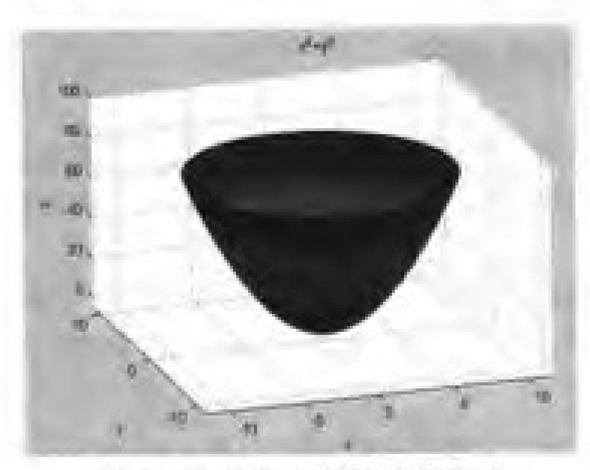


图 6.67 使用三维简易命令绘图



在三维简易命令 ezsuf 中，当选用 circ 参数时，表示指定图形在“圆域”上绘制。相应的圆域是极坐标系，该命令可以十分方便地绘制圆域中的图形。

6.8 图形窗口

除了在 MATLAB 命令窗口中输入命令来创建图形之外，还可以借用 MATLAB 提供的图形窗口。这是一个交互式的窗口，可以很方便地编辑各种图形。MATLAB7.0 相对于之前的 6.x 版本，在图形窗口的功能上有了较大的改进，用户可以在该图形窗口界面上实现几乎所有的编辑功能。

在命令窗口中输入绘图命令后，MATLAB 就会自动调用图形窗口，因此用户也许并不了解该窗口的创建方法以及作用。为此，在本节的第一小节中将介绍如何创建和控制图形窗口，在后面的小节中将介绍如何利用图形窗口编辑 MATLAB 的图形。

6.8.1 创建和控制图形窗口

在 MATLAB 中，创建图形窗口的命令是 figure。该命令的常见调用格式如下：

- figure 创建一个图形窗口对象；
- figure('PropertyName',PropertyValue,...) 按照用户自定义的属性来创建一个图形窗口对象，用户可以对该图形窗口设置相应的属性；
- figure(h) 如果图形句柄 h 已经存在，则该命令会使得该图形窗口成为当前窗口；如果图形句柄 h 不存在，则创建一个句柄值为 h 的图形窗口对象；
- h = figure(...) 返回图形窗口对象的句柄。

如果希望了解当前或者已知图形句柄的窗口信息，可以使用下面的命令：

- get(n) 返回句柄值为 n 的图形窗口的参数名称以及当前数值；
- set(n) 返回可为句柄值为 n 的图形窗口的参数名称，以及用户可为这些参数设置的数值。

例 6.58 在 MATLAB 中，使用相应的命令创建一个图形窗口对象。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>>figure(3)
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 6.68 所示。



图 6.68 创建图形窗口对象



在默认情况下，如果用户只是输入命令"figure"，则会创建名为"Figure 1"的图形窗口。而在上面的程序行中输入的命令为"figure(3)"后，创建的窗口名称则为"Figure 3"。

例 6.59 在 MATLAB 中，使用相应的命令查看上面步骤中创建的图形窗口属性。

step 1 在 MATLAB 中的命令窗口中输入"get(3)"，得到下面的窗口信息。

```
BackingStore = on
CloseRequestFcn = closereq
Color = [ 0.8 0.8 0.8]
...............
PaperPosition = [ 0.634517 6.34517 20.3046 15.2284]
PaperPositionMode = manual
PaperSize = [ 20.984 29.6774]
....................
Type = figure
UIContextMenu = []
UserData = []
Visible = on
```

step 2 在 MATLAB 中的命令窗口中输入"set(3)"，得到下面的窗口信息。

```
Alphamap
BackingStore: [ {on} | off ]
CloseRequestFcn: string -or- function handle -or- cell array
................
Tag
UIContextMenu
UserData
Visible: [ {on} | off ]
```



限于本书的篇幅，上面两段程序代码中都省略了一些图形窗口的信息，读者可以自行在MATLAB命令窗口中演示相应的命令行代码。由于在上面的步骤中，只是创建了基础图形窗口，没有设置其他的窗口信息，因此上面程序描述的就是默认窗口信息。

6.8.2 使用工具栏编辑图形

相对于MATLAB6.x以及之前更新的版本，MATLAB7.0对图形窗口的工具栏进行了较大幅度的改进。其中，最显著的改进之处在于增加了移动图形、在图形中动态取点以及为图形中的数据点增加pin等功能。

例 6.60 在 MATLAB 中，使用简单的函数实例来演示 MATLAB 新增的功能。

step 1 在 MATLAB 中绘制一个新的函数图形，该函数的源数据是一个 100 × 1 的随机数值矩阵，由 MATLAB 中的内置函数 rand 产生。可以使用 MATLAB7 中新增的工具栏来绘制该函数图形，如图 6.69 所示。

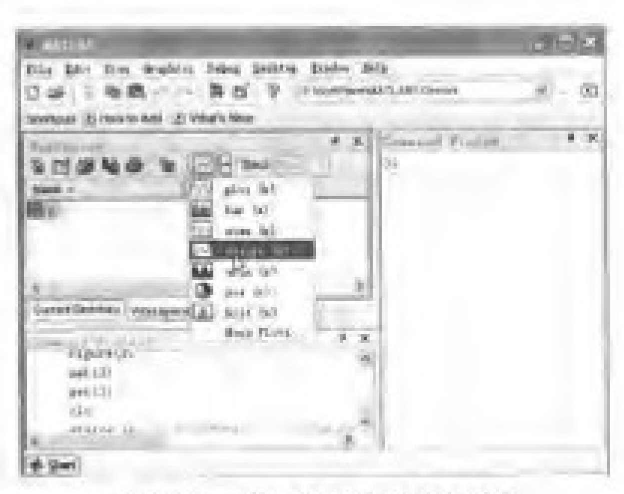


图 6.69 利用窗口菜单创建图形

在上面的图形中，使用了“工作空间”中的绘图按钮来直接绘制变量 x 的图形，当选择相应的图形类型后，MATLAB 会自动创建图形，如图 6.70 所示。

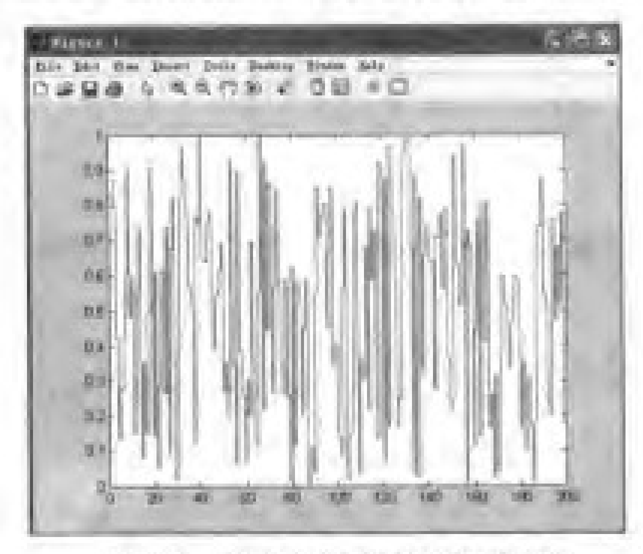


图 6.70 创建图形窗口以及窗口图形

step 2 查看局部数据变化情况。可以选择图形窗口工具栏中的“放大”按钮，然后选择需要放大的局部数据，在图形窗口中划出放大的范围，如图 6.71 所示。

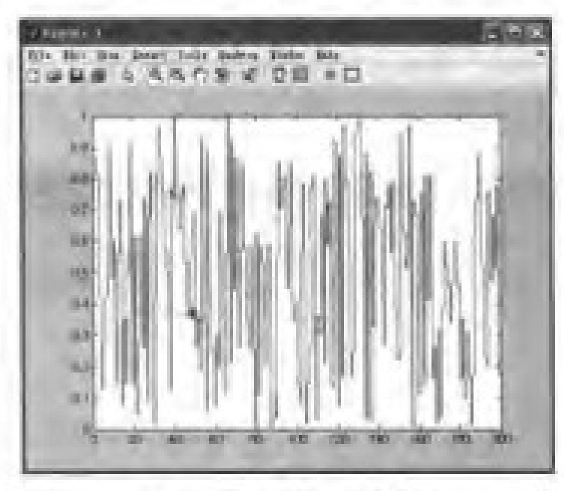


图 6.71 放大局部数据

当松开鼠标后，图形窗口显示放大后的局部图形，在放大后的图形中编辑相应的图形，如图 6.72 所示。

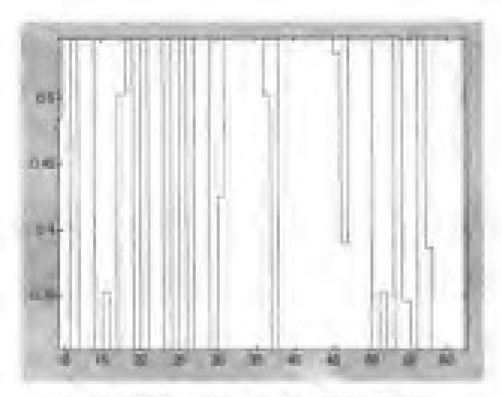


图 6.72 放大后的局部数据

step 3 移动图形，查看不同的数据范围。由于在上面的步骤中放大了局部数据，图形中 X 坐标轴的刻度范围变小，如果需要查看其他数据范围的函数情况，就需要在图形窗口中移动图形。选择工具栏中的按钮，移动图形，如图 6.73 所示。

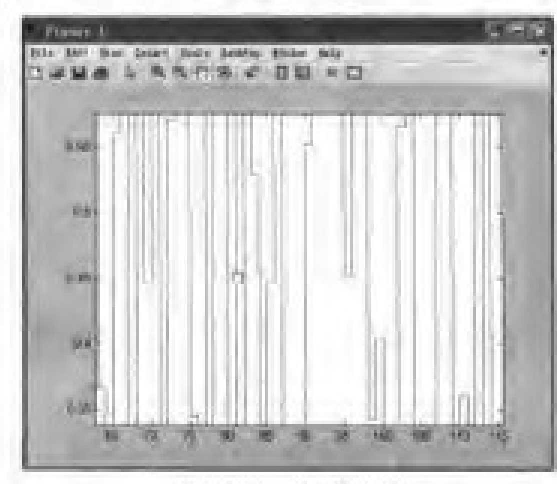


图 6.73 移动图形



当移动图形的时候，图形窗口中会显示"手形"的移动图标，表示图形处在移动当中。同时，图形窗口中的X轴和Y轴刻度范围会自动调整，实现最好的显示结果。

step 4 查看图形数据点的坐标值。选择工具栏中的取点按钮，然后在图形窗口中选择图形中的数据点，查看该数据点的坐标值，如图 6.74 所示。

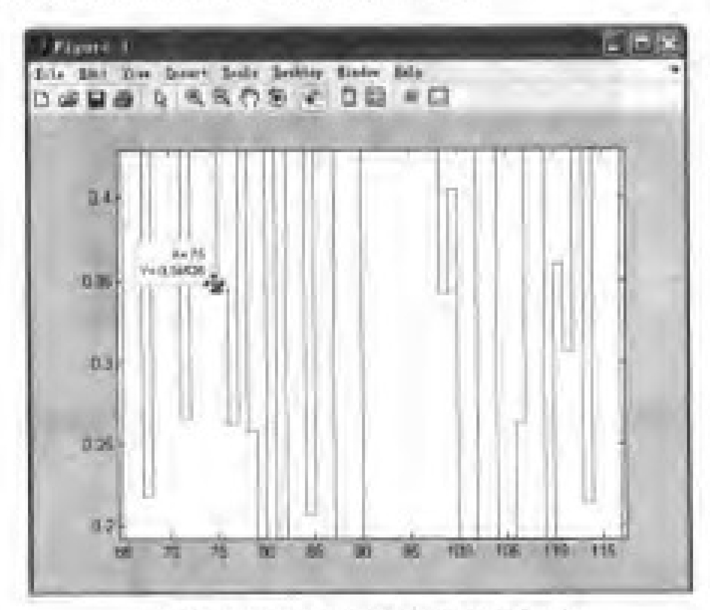


图 6.74 查看图形数据点的坐标值



当在图形中选择数据点后，MATLAB图形中就会显示数据点的坐标值；只要是函数图形中的任何数据点，都可以很方便地查看该点的坐标值。

step 5 修改坐标值的显示方式。上面图形中显示的是默认方式，用户可以修改坐标值的显示方式。选中上面步骤中显示的坐标值框，然后单击鼠标右键，在弹出的快捷菜单中选择 "Display style" ⇨ "Window Inside Figure" 选项，如图 6.75 所示。

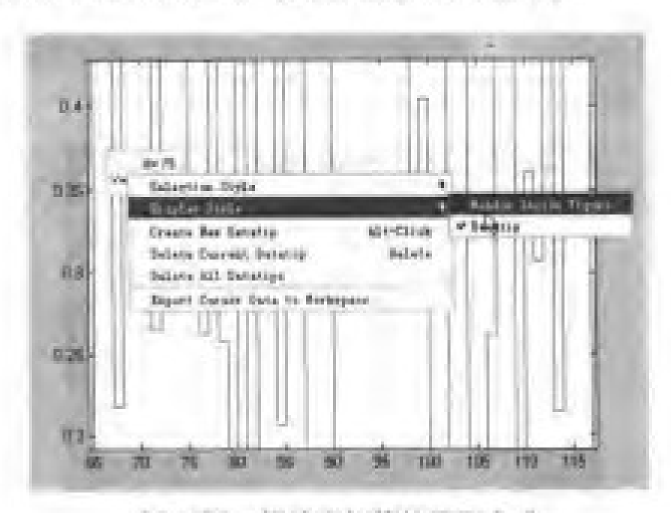


图 6.75 修改坐标值的显示方式

当选择相应的选项后，可以查看修改后的坐标值显示方式，如图 6.76 所示。

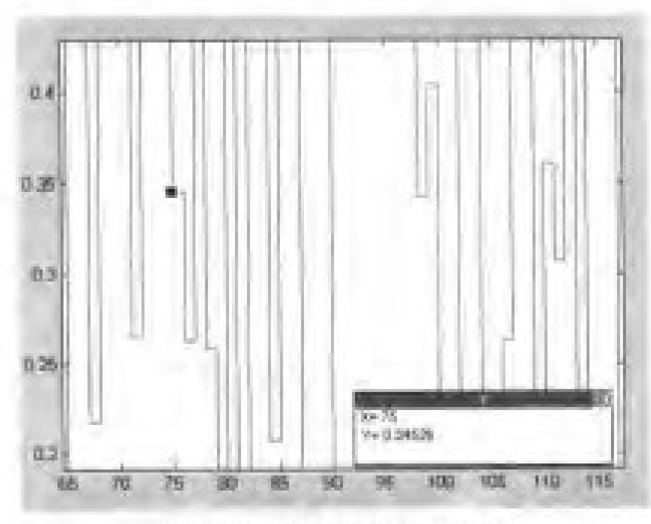


图6.76 修改后的显示方式



当确定坐标值的显示方式后，再次单击图形窗口工具栏中的“取点”按钮，就可以完成坐标值的显示方式。为了显示图形窗口的其他工具按钮，在本步骤中需要删除该坐标值数值。

step 6 添加图形的注释文件。在前面曾经使用annotation命令来添加图形注释，其实使用图形窗口可以很方便地完成这样的工作。选择图形窗口工具栏中的“添加文字箭头”按钮，然后在图形中设置注释文字，如图6.77所示。

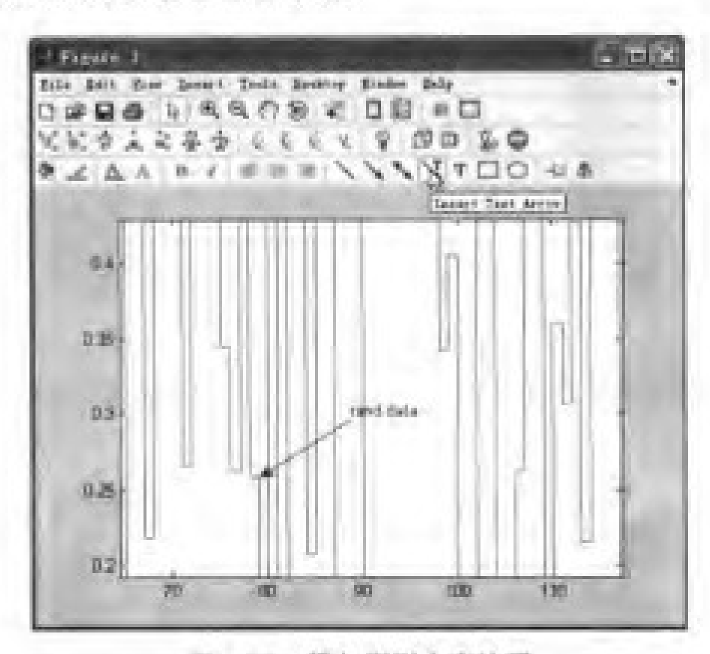


图6.77 添加图形文字注释



为了更好地解释图形窗口中的工具栏按钮，可以选择工具栏中的“View”下拉菜单中的前面三个菜单选项，显示图形窗口的所有工具栏按钮。

step 7 移动图形窗口，查看图形注释文件。上面步骤中添加了图形的文字注释，可以移动图形窗口来查看注释文件的变化，如图6.78所示。

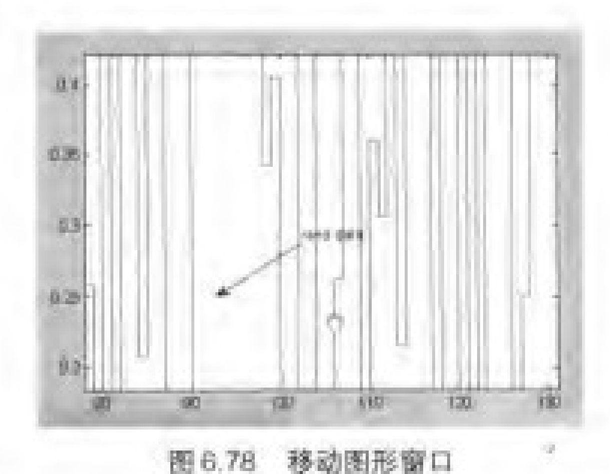


图 6.78 移动图形窗口

从上面的图形窗口中可以看出，当移动图形窗口时，文字注释并不会随着图形移动，而固定在图形注释文件的插入点上。

step 8 固定注释文件。选择上面步骤中添加的文字注释，然后选择图形窗口中的 "pin to axes" 按钮，再选择注释文件的固定点，如图 6.79 所示。

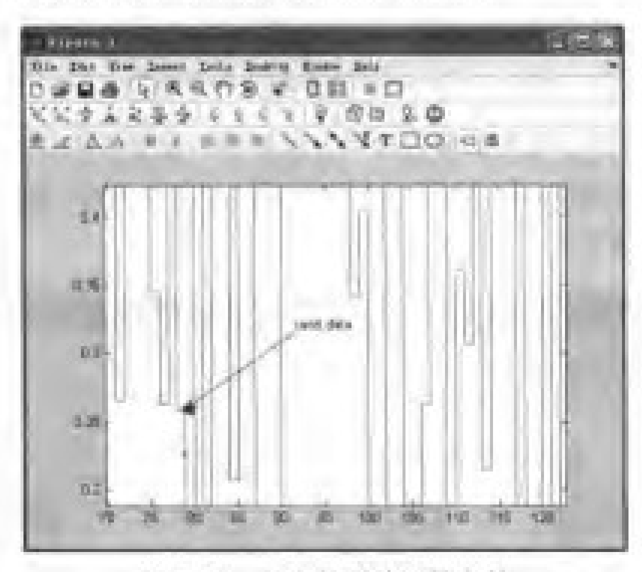


图 6.79 固定图形的注释文件

当固定图形中的注释文字后，再次移动图形窗口，就可以查看文字注释已经固定在原来的数据点上，如图 6.80 所示。

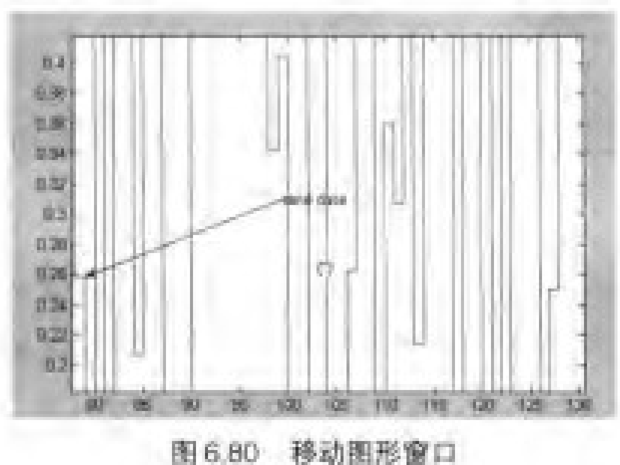


图 6.80 移动图形窗口

6.8.3 使用绘图工具（plot tool）编辑图形

MATLAB7.0 在图形窗口中增加了绘图工具(Plot Tool)组件，该组件由图形窗口面板（Figure Palette）、绘图浏览器（Plot Browser）和属性编辑面板（Property Editor）组成。在默认情况下，该绘图工具是隐藏的，如果希望调用该绘图工具，可以单击图形窗口中对应的显示按钮。在本小节中，通过使用一个简单的图形实例来说明如何调用和使用绘图工具。

例 6.61 在 MATLAB 中，绘制函数 peaks 的三维图形，然后使用图形窗口中的绘图工具来编辑该三维图形的属性。

step 1 在 MATLAB 中的命令窗口中输入下面的命令，创建 peaks 函数的三维图形。

```
>> z=peaks(45);
>> surf(z)
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 6.81 所示。

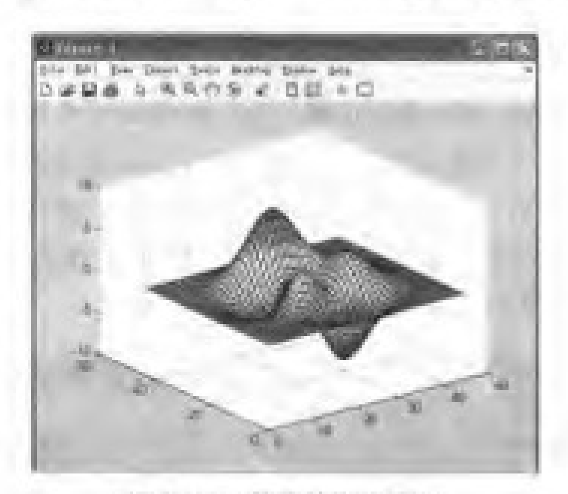


图 6.81 创建的图形窗口

step 3 单击图形窗口中的"Show Plot Tool"按钮，打开绘图工具（Plot Tool）组件。由于绘图工具在默认情况下是隐藏的，因此需要调用绘图工具组件，如图 6.82 所示。

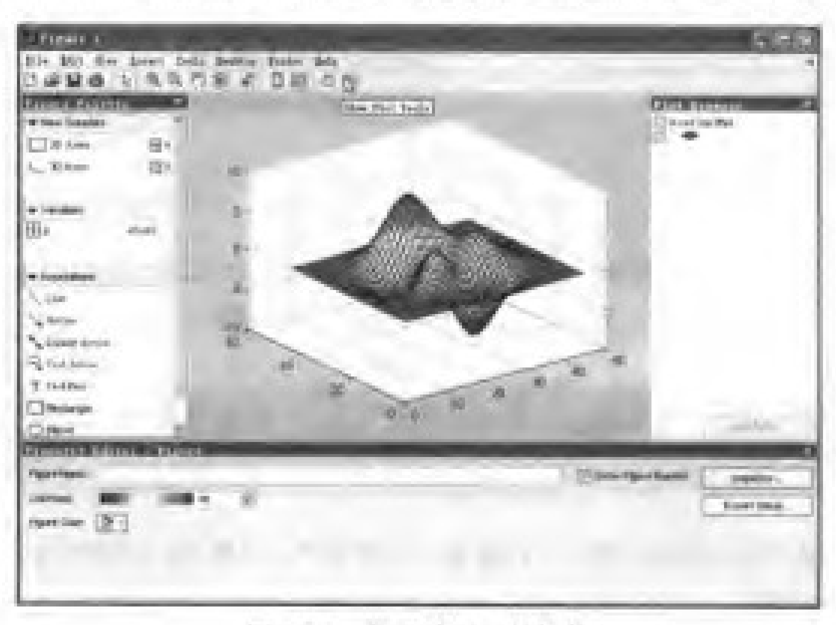


图 6.82 打开绘图工具组件

从图形窗口中可以看出，绘图工具的三个组件分别列在图形窗口的左、右侧和底部。其中图形窗口面板（Figure Palette）列在图形窗口的左侧，绘图浏览器（Plot Browser）列在图形窗口的右侧，属性编辑面板（Property Editor）列在图形窗口的底部。下面简要介绍这些面板在编辑图形中所起的作用。

- **图形窗口面板（Figure Palette）**：在该图形编辑面板中，可以为图形窗口添加新的子图（New Subplots）。这个子图不受维度限制，可以添加二维或者三维子图。在该编辑面板中，还可以为图形窗口添加各种形式的注释（annotation）。最后，可以在该面板中查看图形的变量。
- **绘图浏览器（Plot Browser）**：在该面板中，可以查看图形中的各个对象，例如坐标轴、变量或者整个图形对象等。同时，可以单击该面板中的"Add Data"按钮，为图形中添加新的数据。
- **属性编辑面板（Property Editor）**：该面板将根据用户在图形窗口选择的对象显示不同的属性列表，用户可以在该面板中设置选中的图形对象的各种属性。这是最有有效的一个工具，可以避免繁杂的程序代码而达到编辑图形的目的。

step 4 修改坐标轴的刻度间隔。在绘图浏览器（Plot Browser）中选择"Axes"对象，属性编辑面板（Property Editor）中就会显示图形中所有的坐标轴对象。选择"X Axis"选项卡，单击"Ticks..."按钮，打开"Edit Axes Ticks"对话框，选择其中的"X Axis"选项卡，然后选择"Step by"单选钮，在其后输入新的间隔"5"，单击"Apply"按钮，就可以修改图形中的X坐标轴的刻度间隔，如图 6.83 所示。

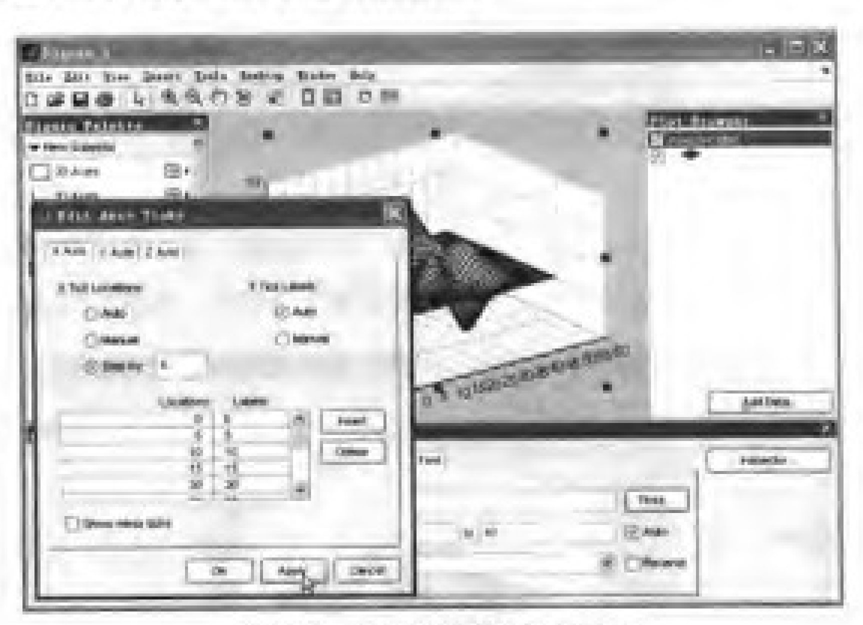


图 6.83　设置坐标轴的刻度间隔



在前面介绍过，关于图形坐标轴的属性设置，一般需要借用图形的底层命令。但是，使用图形窗口的绘图工具可以很方便的完成关于坐标轴的属性设置。

step 5 修改其他坐标轴的刻度间隔。可以重复上面的步骤，设置Y坐标轴的刻度间隔，修改后的图形如图 6.84 所示。

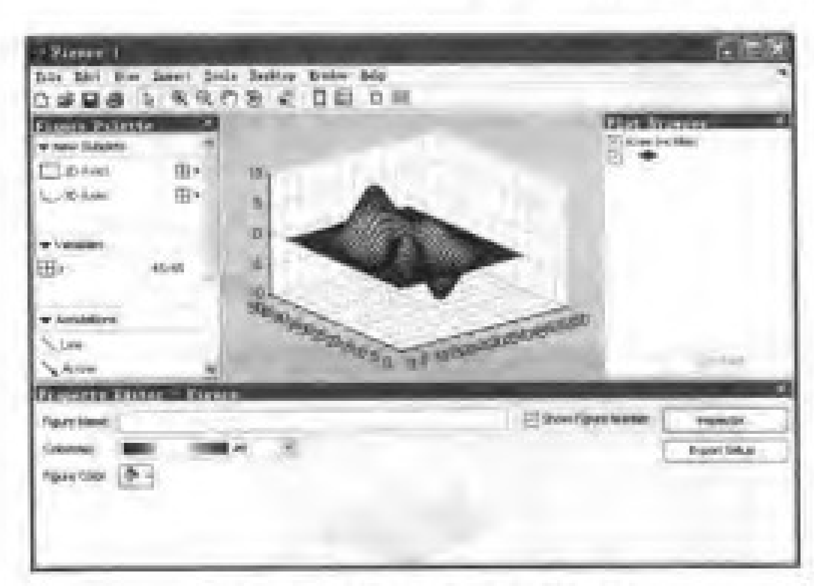


图 6.84 修改坐标轴间隔后的图形



当修改坐标轴的刻度间隔后，X坐标轴和Y坐标轴刻度相对于原来图形中的刻度间隔更密，对应的分格线也显得更详细。

step 6 修改坐标轴的显示比例。在绘图浏览器（Plot Browser）中选择"Axes"对象，在属性编辑面板（Property Editor）中选择"X Axis"选项卡，选择"X Scale"选框中的"log"，同时选中"Reverse"选项，得到的图形如图 6.85 所示。

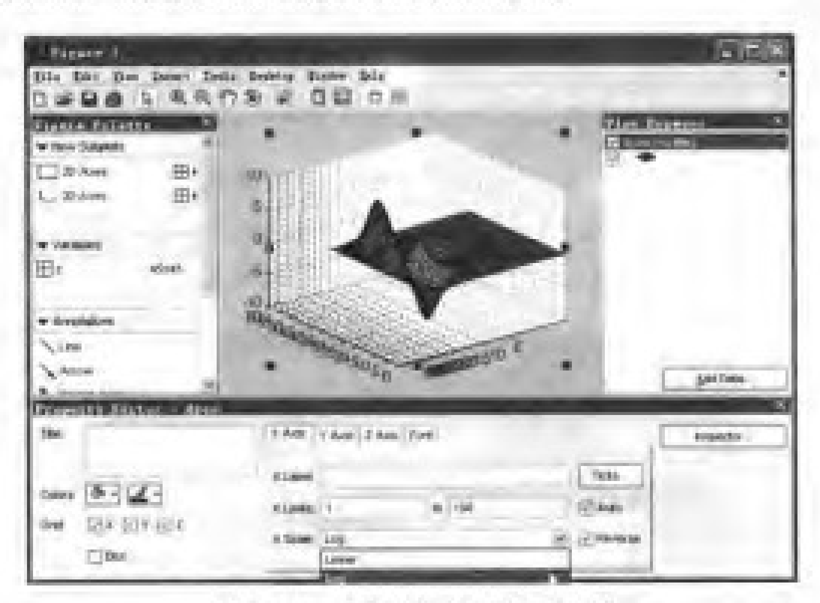


图 6.85 修改坐标轴的显示比例

上面的操作相当于将X坐标轴设置为对数，并且反向显示。如果需要使用MATLAB的命令实现上面的结果，都需要使用相关的底层命令。



由于上面这个图形比较简单，所以仅仅演示了使用绘图工具编辑图形坐标轴的用法。对于图形中其他对象，都可以使用相同的方法来设置属性。

step 7 添加新的变量。由于上面的图形中只有一个变量z，为了在后面的步骤中添加新的图形对象，在本步骤中需要添加新的变量。切换到MATLAB的命令窗口中，然后输入下列的程序代码。

```
>> x=[0:0.01:20*pi];
>> y=cos(2*x)+sin(x);
```

step 8 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图6.86所示。

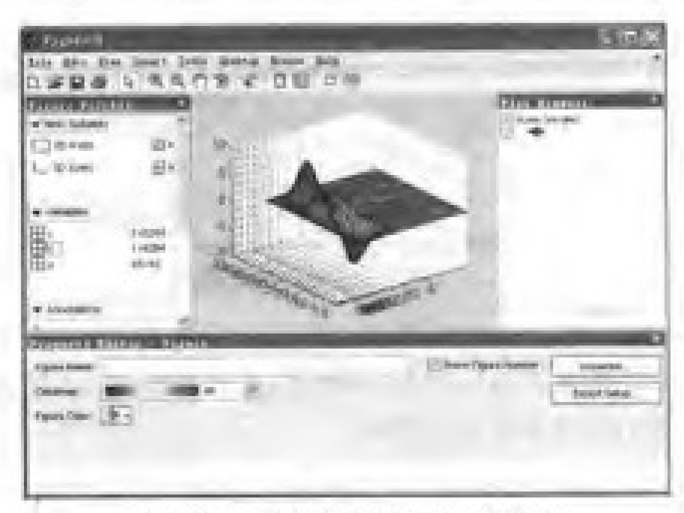


图6.86　添加新变量后的图形窗口



当在MATLAB内存中添加新的变量后，在图形窗口的图形窗口面板（Figure Palette）中显示添加的变量，就像MATLAB中的工作空间一样。这样，可以很方便地为添加所有的变量绘制和编辑图形。

step 9 向上面的图形中添加新的变量。选中上面的图形，绘图浏览器（Plot Browser）中的"Add Data..."按钮会被激活。单击"Add Data..."按钮，弹出"Add Data to Axes"对话框，在该对话框中，需要首先选择添加变量的图形类型，如图6.87所示。

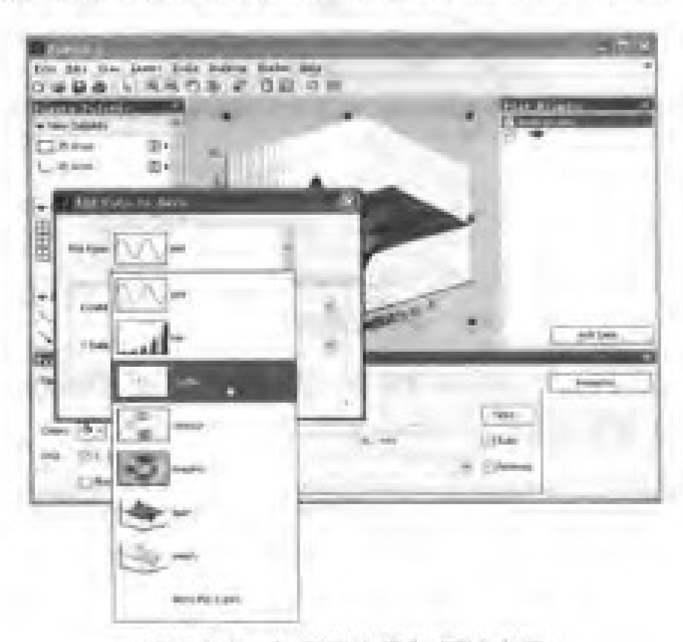


图6.87　向图形中添加新的变量



在"Add Data to Axes"对话框中提供MATLAB中所有可行的图形类型，可以根据需要选择所需的图形类型，在本例中选择的是"Scatter"图形类型。

step 10 选择添加图形的数据来源。在"Add Data to Axes"对话框中，为所选择的图形类型添加数据来源，本例添加的数据来源就是前面步骤设置的变量x和y，如图6.88所示。



图 6.88 向图形中添加新的变量

当为图形选择新的图形变量后，单击"Add Data to Axes"对话框中的"OK"按钮，MATLAB就会在图形中添加该变量的散点图，如图6.89所示。

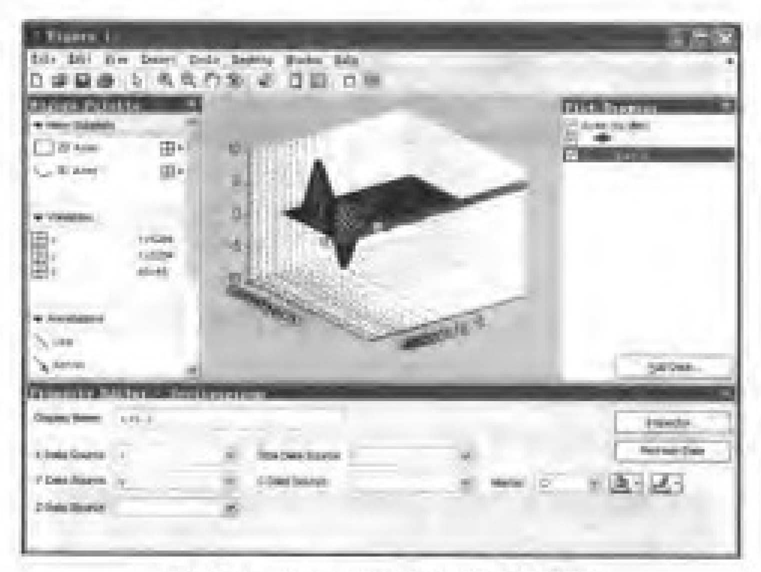


图 6.89 在图形中显示添加的变量



当添加新的变量后，在默认情况下MATLAB的绘图浏览器（Plot Browser）会自动选中添加的变量图形，因此在属性编辑面板（Property Editor）中会显示对应的属性。

step 11 添加新的子图。在前面的步骤中为图形添加了新的变量，在绘图工具中还可以添加新的子图。选择图形窗口面板（Figure Palette）中的"New Subplots"选项下的"2D Axes"选框中的添加子窗口，如图6.90所示。

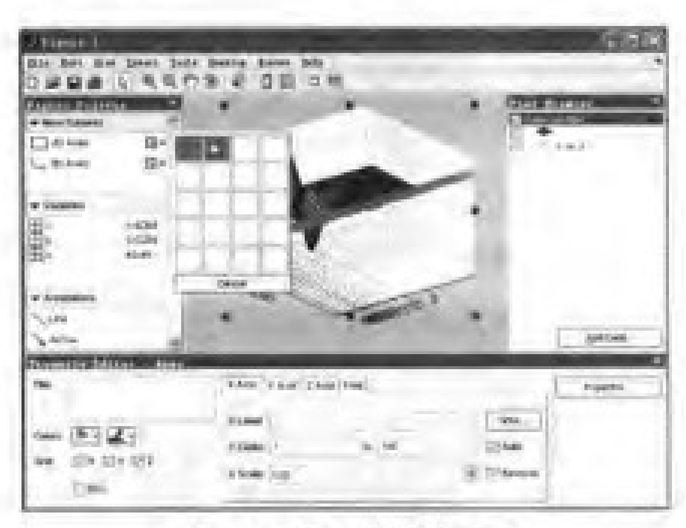


图 6.90　在图形中添加子图

在上面步骤所打开的窗格中，可以选择子图的格式，如果选择纵向的方格，则会添加纵向的子图。而在本例中，选择添加的是横向的子图，得到的结果如图 6.91 所示。



图 6.91　在图形中添加新的坐标系

step 12 添加子图的数据。使用步骤(7)中的方法为新坐标轴添加新的数据，在本步骤中选择的图表类型是“plot”，图表的数据来源则是 x 和 y，如图 6.92 所示。

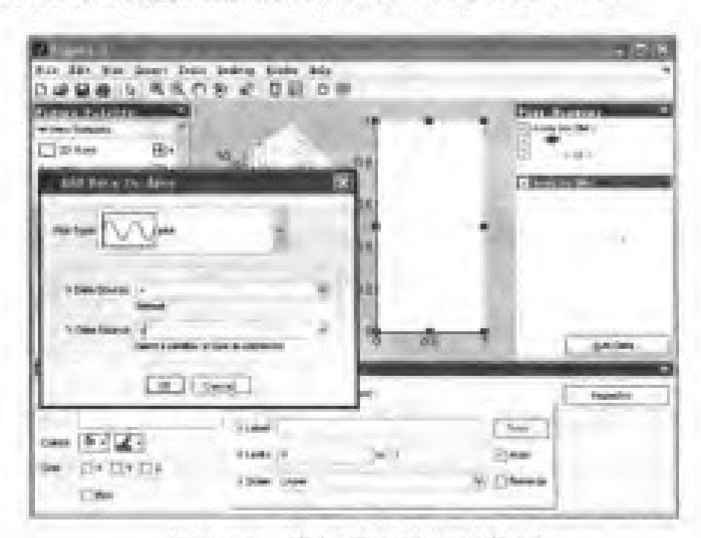


图 6.92　添加子图的图形数据



由于在上面的步骤中添加了新的坐标系，因此在本步骤中添加新数据时，需要首先选中对应的坐标系，否则就相当于在原来的坐标系中添加变量。

step 13 编辑上面步骤中添加的子图。单击 "Add Data to Axes" 对话框中的 "OK" 按钮，查看上面步骤中完成的子图，如图 6.93 所示。

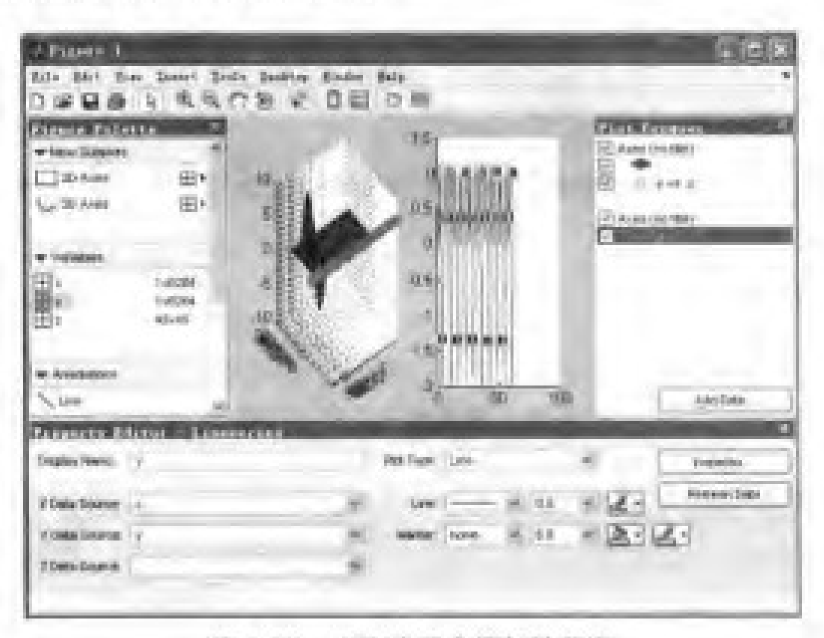


图 6.93 上面步骤中添加的子图

在默认情况下，MATLAB 会选中添加的线形曲线。为了编辑坐标轴的范围，需要选中图形的坐标轴，然后在属性编辑面板中修改 X 轴的刻度范围，如图 6.94 所示。

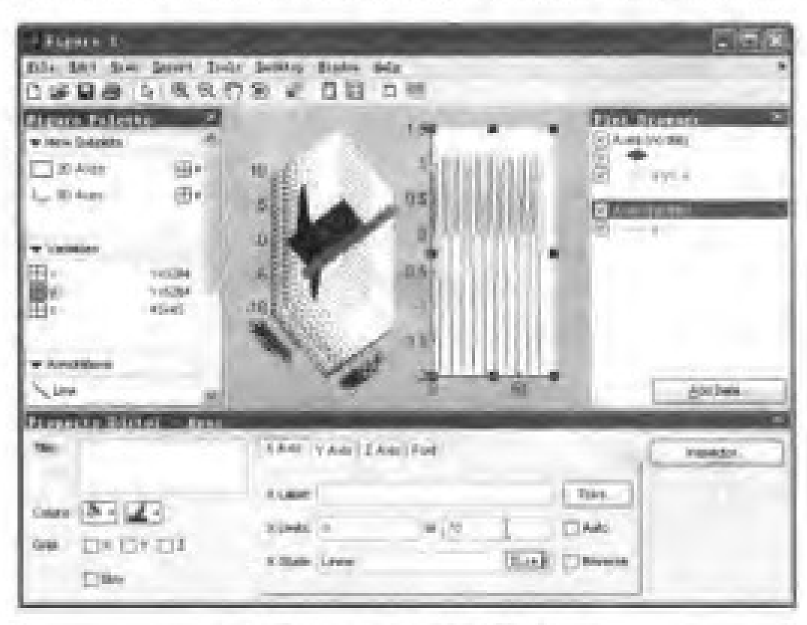


图 6.94 修改坐标轴的刻度范围

step 14 修改子图的显示方式。由于上面图形的显示方式并不能很好地查看图形，因此需要修改子图的显示方式。选择图形窗口面板（Figure Palette）中的 "New Subplots" 选项下的 "2D Axes" 选框中的添加子窗口，如图 6.95 所示。

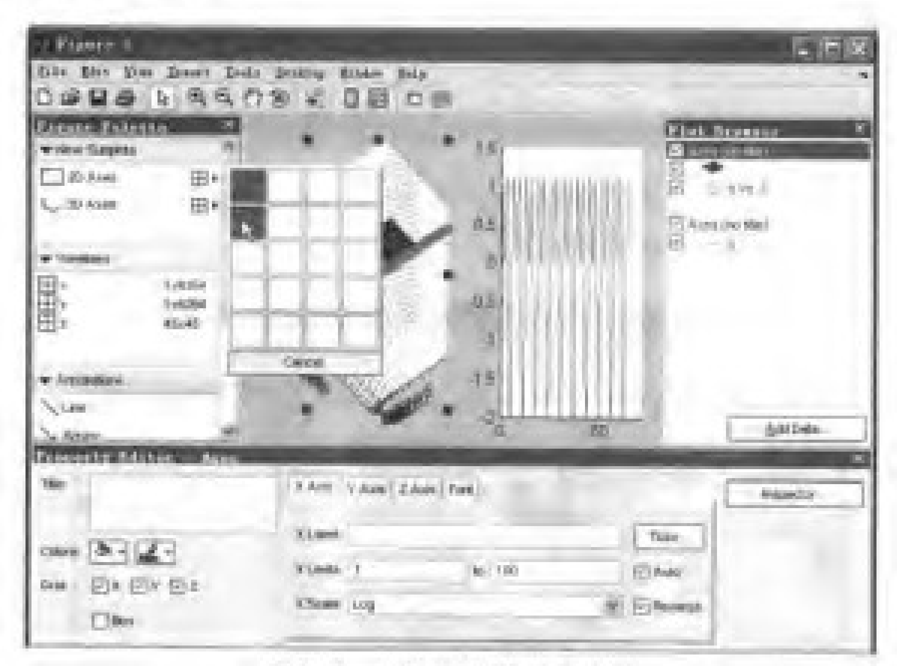


图 6.95　修改子图的显示方式



如果在命令窗口中修改子图的属性，一般需要使用MATLAB中的底层绘图命令，而在图形编辑窗口只需要使用一个简单的选项就可以了。

step 15 查看修改后的子图。当选择子图显示方式的选项后，可以查看修改后的子图，如图 6.96 所示。

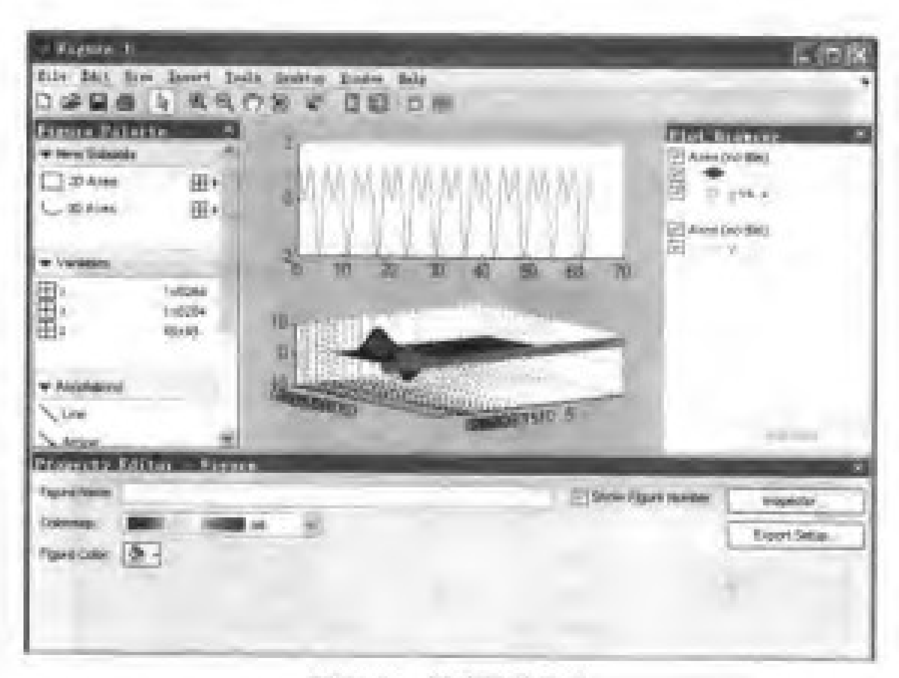


图 6.96　修改后的子图

step 16 修改第一个子图的Y轴坐标范围。参考步骤(3)，修改第一子图的坐标轴的刻度和坐标轴范围，得到的结果如图 6.97 所示。

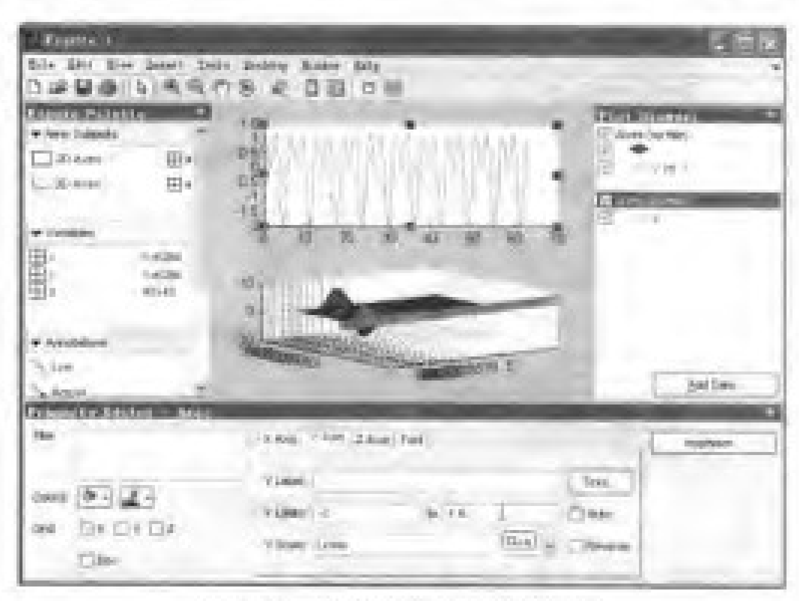


图 6.97 修改子图的坐标轴范围

step 17 设置第一个子图的标题、边框和分格线。为了更加直观地查看图形，可以为第一个子图设置标题、边框和分格线，在属性编辑面板（Property Editor）中选中相应的选项，得到的结果如图 6.98 所示。

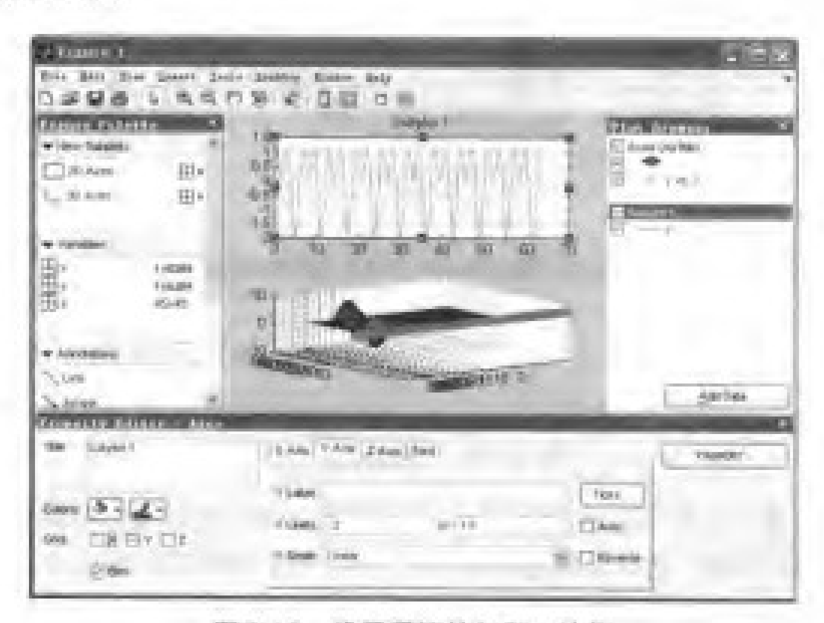


图 6.98 设置子图的标题、边框

step 18 查看图形的其他属性。前面已经介绍图形工具在图形编辑中的常见应用，如果需要编辑图形的其他属性，可以选择相应的图形，然后单击属性编辑面板（Property Editor）中的 "Inspector..." 按钮，打开 "Property Inspector" 对话框，可以在其中设置图形对象的属性，如图 6.99 所示。



可以在上面步骤中打开的 "Property Inspector" 对话框中，设置该图形的所有相关属性。限于空间限制，属性编辑面板（Property Editor）列出的仅仅是常见属性选项。



图 6.99　查看图形的其他属性

step 19 隐藏绘图工具。绘图工具的主要功能是编辑图形，当完成图形的编辑工作后，为了得到最后的图形，需要隐藏绘图工具。单击图形窗口中的 "Hide Plot Tools" 按钮，隐藏绘图工具，如图 6.100 所示。



图 6.100　隐藏绘图工具

当单击图形窗口中的 "Hide Plot Tools" 按钮后，整个图形窗口中就会只有图形，常见的编辑窗口都会隐藏，适当编辑图形窗口的大小，得到的图形如图 6.101 所示。

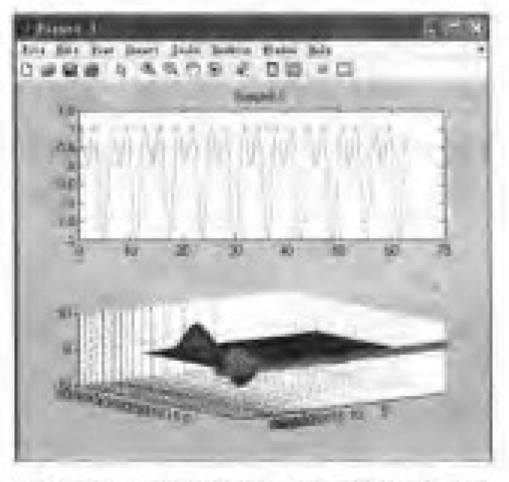


图 6.101　隐藏绘图工具后的图形窗口

6.8.4 使用图形窗口进行数据分析

在介绍数据分析的章节中，了解到图形工具在数据分析中的重要作用。基于这种原因，MATLAB 在图形窗口增加进行数据分析的菜单选项。

例 6.62 使用 6.8.3 小节中的图形进行数据分析。

step 1 打开数据分析的对话框。选中图6.96中的第一个子图，然后选择菜单栏中的 "Tool" ⇨ "Data Statistics" 命令，如图6.102所示。

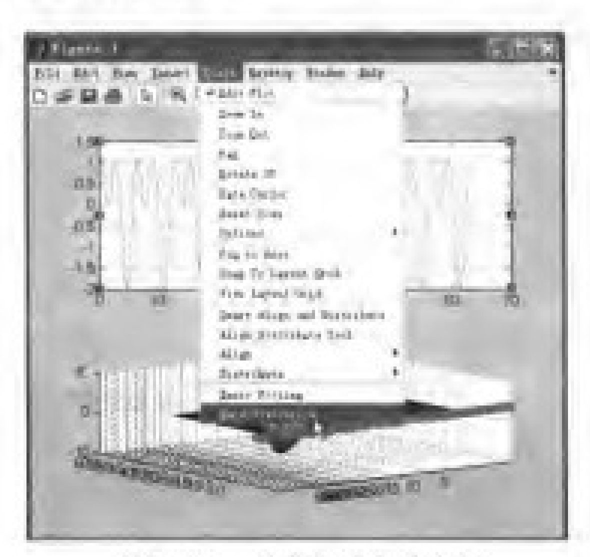


图 6.102 选择数据分析选项

step 2 选择数据统计选项。当选择相应的选项后，系统会出现 "Data Statistics-1" 对话框。该对话框中显示该图形中变量 x 和 y 的最小值、最大值和平均值等，选择 x 和 y 的平均值，得到的图形如图 6.103 所示。

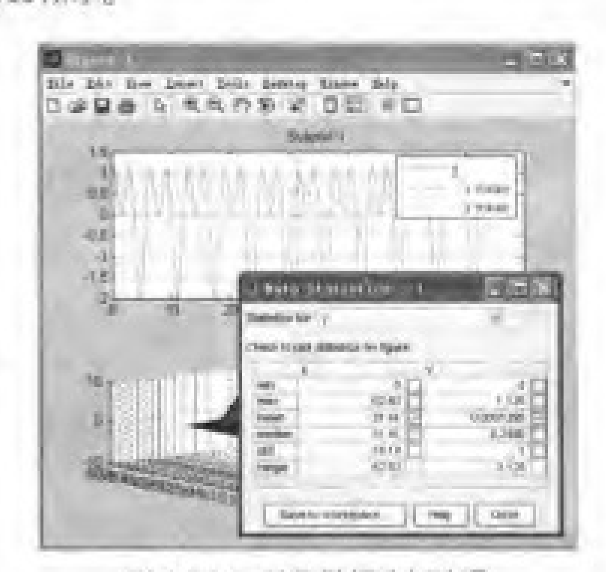


图 6.103 选择数据分析选项

当选择相应的变量参数后，MATLAB 自动在图形中添加该参数数值的直线，且会显示相应的图例。这样，就可以在第一个子图中直观地查看相应的统计信息。



如果需要知道这些函数的统计信息，可以使用MATLAB中相关的统计函数，再使用绘图命令将这些统计信息绘制在原来的图形中。图形窗口中对应的选项则简化了相应的操作过程，只需要在对话框中选择相应的选项就可以了。

step 3 对图形中的数据进行拟合。选中图形中的第一个子图，然后选择菜单栏中的"Tool"⇨"Basic Fitting"命令，如图6.104所示。

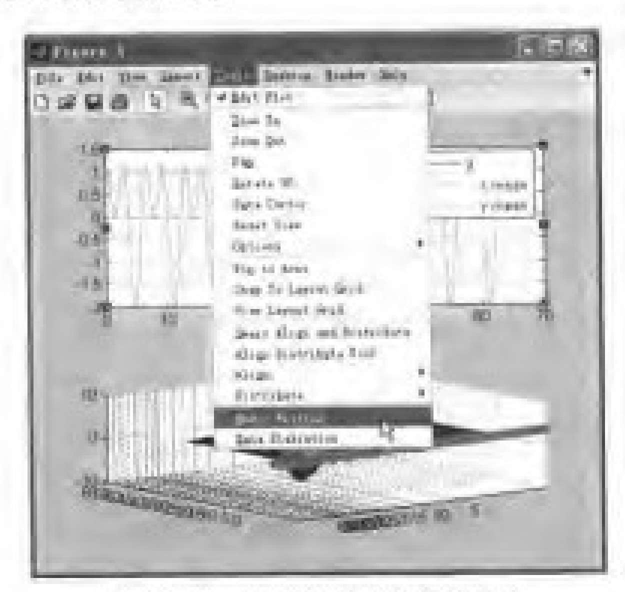


图6.104 选择图形中的数据拟合

step 4 设置数据拟合的参数。当选择"Basic Fitting"命令后，MATLAB调用"Basic Fitting-1"对话框，可以在该对话框中设置拟合的参数，如图6.105所示。



图6.105 设置数据拟合的参数

在"Basic Fitting-1"对话框中，可以设置拟合的选项，本例中选择"10th degree polynomial"选项，进行10阶拟合。然后选中"Show equations"选项，表示在拟合的曲线中显示拟合的方程。最后，选中"Plot residual"选项，表示绘制拟合的误差曲线。设置拟合参数后的基础图形，如图6.106所示。

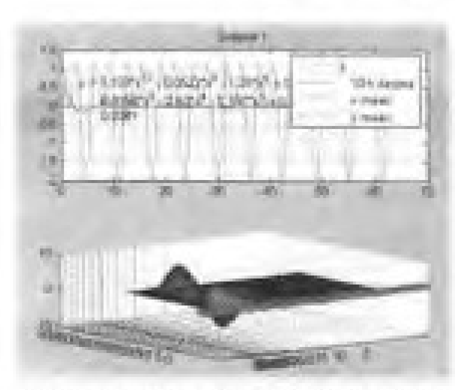


图 6.106 设置拟合参数后的曲线

step 5 查看拟合的误差曲线。在前面的步骤中，选择了绘制拟合的误差曲线。查看该曲线，如图6.107所示。

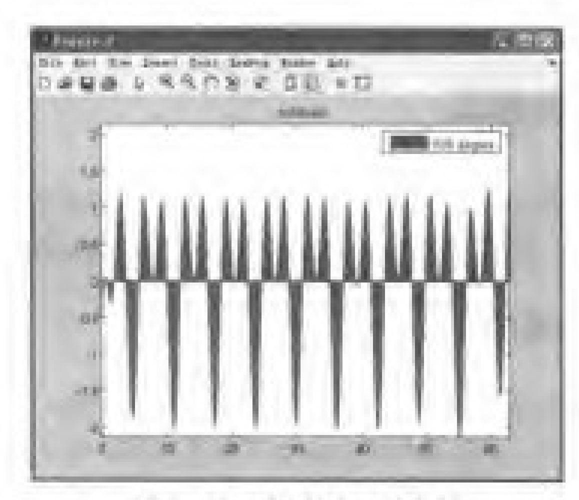


图 6.107 拟合的误差曲线



由于在前面步骤中绘制了一个多子图的图形，因此MATLAB会在一个新的图形窗口"Figure2"中显示误差曲线。如果只绘制一个图形，则会在同一个图形窗口显示该误差曲线。

step 6 创建图形的M文件。返回到"Figure1"图形窗口中，选择菜单栏中的"Tool" ⇨ "Generate M-files..."命令，如图6.108所示。

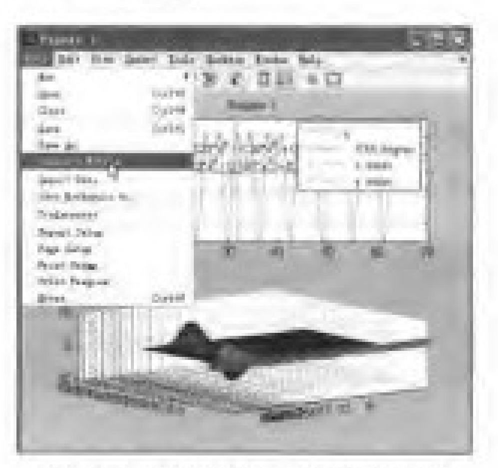


图 6.108 创建绘制图形的M文件

当选择相应的菜单命令后，MATLAB 会自动创建图形的 M 文件，如图 6.109 所示。



图 6.109　对应的 M 文件

6.9 绘制复数变量图形

在 MATLAB 中，为用户提供绘制复数变量图形的一些函数。前面介绍过，对于复数变量，MATLAB 可以绘制相应的实部和虚部，但是，MATLAB 可以根据复数的结构绘制更加复杂的图形。

6.9.1 绘制复数图形原理

MATLAB 提供的绘制函数包括 CPLXMAP、CPLXGRID 和 CPLXROOT 等，这些函数名称的前缀 CPLX 是英文单词 COMPLEX（复数）的缩写。这些函数的功能尽管各不相同，但是对于所绘制的图形有如下共同的特点。

假定 f(z)是某一函数，函数的变量是复数，使用 CPLXMAP、CPLXGRID 或者 CPLXROOT 命令绘制的三维曲面图将会以函数的实部为高度，以其虚部为颜色。在默认的情况下，这些颜色的变化范围是 HSV 颜色模式。

在 MATLAB 中，函数 CPLXGRID 将产生一个极坐标下的复数网格数据点，例如函数表达式 CPLXGRID (m)将产生一个$(m-1)\times(2m-1)$的复数极坐标网格数据点。该命令和前面的 meshgrid 函数类似，功能是产生数据格点，只是这些数据格式都是复数的，而不是实数的。其对应的 M 文件如下：

```
function z = cplxgrid(m)
r = (0:m)'/m;
theta = pi*(-m:m)/m;
z = r * exp(i*theta);
```

可以看出，该函数产生的复数极径的数值范围是[0, 1]，极角的范围是$[-\pi,\pi]$，然后用极坐标的方法创建复数矩阵 z，而且矩阵维度是$(m-1)\times(2m-1)$。



在使用后续函数来绘制复数图形时，该函数绘制复数图形的基础函数，用来产生绘制图形的数据系列点。关于使用极坐标方法创建复数的方法，在第 1 章中就有介绍，不太熟悉的读者可以查阅相应的内容。

6.9.2 CPLXMAP 命令

CPLXMAP 命令的功能是绘制复数函数的图形，其语法是 CPLXMAP(z,f(z),bound)。其中，参数 z 是定义域，f(z)是产生图形区域的函数。为了更好地解释该函数的功能，下面调用该函数的 M 文件：

```
function cplxmap(z,w,B)
blue = 0.2;
x = real(z);
y = imag(z);
u = real(w);
v = imag(w);
if nargin > 2
   k = find((abs(w) > B) | isnan(abs(w)));
   if length(k) > 0
      u(k) = B*sign(u(k));
      v(k) = zeros(size(k));
      v = v/max(max(abs(v)));
      v(k) = NaN*ones(size(k));
   end
end
M = max(max(u));
m = min(min(u));
axis([ -1 1 -1 1 m M]);
caxis([ -1 1]);
s = ones(size(z));
mesh(x,y,m*s,blue*s);
hold on
surf(x,y,u,v);
hold off
colormap(hsv(64))
```

在上面的程序代码中，首先求得复数矩阵z和函数数值矩阵w中元素的实部和虚部，然后确定绘制图形的坐标轴范围。在完成数据准备和坐标轴范围后，使用mesh命令绘制曲线图。该曲线图是一个平面图，z 向坐标值就是u的最小值；最后，使用surf命令绘制三维曲面图，该曲面的z向坐标是函数矩阵 w 的实部数值，而函数矩阵 v 的虚部数值则是填充曲面图的颜色矩阵。



上面的程序代码并不复杂，主要是将函数的参数根据图形设置的范围绘制三维曲面线和曲线图。对于其中比较复杂的部分，读者可以不用理解，或者查阅本书中M编程的章节。

6.9.3 CPLXMAP 命令图形实例

例 6.63 使用CPLXMAP 绘制复数函数的三维图形。

step 1 在 MATLAB 命令窗口中输入下面的命令：

```
>> z = cplxgrid(45);
>> cplxmap(z,z.^3)
>> title('z^3')
>> colorbar
```

step 2 查看图形结果。输入代码后，按"Enter"键，得到的图形如图6.110所示。

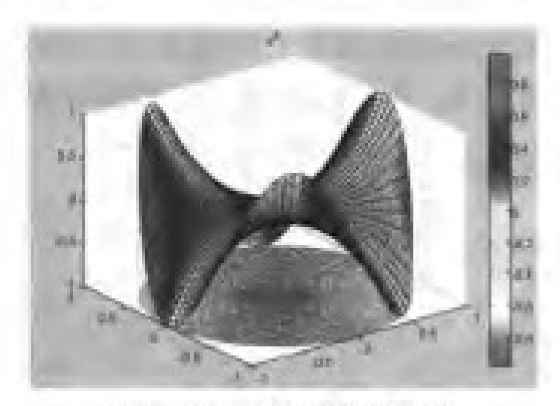


图6.110　完成的复数图形

在上面的图形中，图形的高度就是复数函数$f(z)=z^3$计算结果的实部，而其颜色则是按照计算结果的虚部数值进行染色的。

6.9.4　CPLXROOT 命令

在MATLAB中，还提供CPLXROOT函数来绘制复数函数$f(z)$的n重根对应的的曲面图，其对应的M文件如下：

```
function cplxroot(n,m)
if nargin < 1, n = 3; end
if nargin < 2, m = 20; end
r = (0:m)'/m;
theta = pi*(-n*m:n*m)/m;
z = r * exp(i*theta);
s = r.^(1/n) * exp(i*theta/n);
surf(real(z),imag(z),real(s),imag(s));
```

使用上面的程序代码，可以很轻松地画出比较复杂的复数数值解的曲面图，下面使用一个简单的实例来说明。

6.9.5　CPLXROOT 命令图形实例

例6.64　使用CPLXROOT函数绘制复数立方根的三维图形。

step 1 在MATLAB命令窗口中输入下面的命令：

```
>> z = cplxgrid(45);
>> view(-37.5,30)
>>cplxroot(3)
>>title('z的立方根')
```

step 2 查看图形结果。输入代码后，按"Enter"键，得到的图形如图6.111所示。



上面列举的例子都不复杂，只是为了演示如何使用相应的复数函数来绘制三维图形，可以使用其他的复数来检测这些函数的功能。

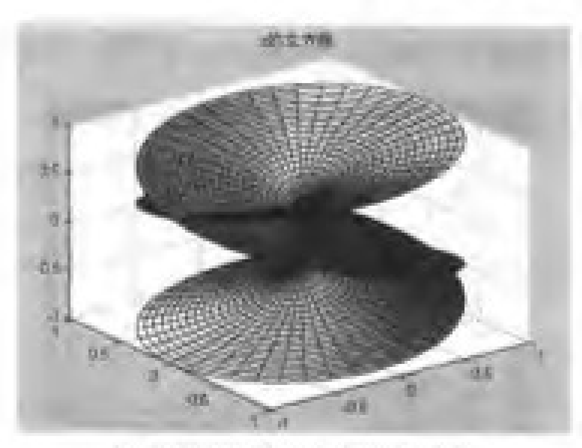


图6.111 复数立方根的图形

6.10 图形的打印和输出

在本章的最后，将简要介绍如何在绘制图形后将完成的图形输出或者打印。这里提到的输出，表示将完成的图形保存在其他文档中，或者使用其他图形软件对该图形进行编辑。在MATLAB中，需要完成图形的打印和输出，可以使用图形窗口的菜单进行操作，也可以使用命令窗口中的命令。

使用菜单方式完成图形的打印和输出，比较简单、直接。使用命令方式完成图形的打印和输出，比较丰富灵活。用户可以在M文件中调用相应的命令，自动执行图形的打印和输出工作。

6.10.1 图形打印的菜单操作方式

为了演示如何使用图形窗口中的菜单进行图形输出，以例6.48完成的图形为例，简单显示基本的操作方式，下面详细介绍相应的操作步骤。

例6.65 演示如何打印例6.48所绘制的三维图形。

step 1 进行页面设置。选择图形窗口中的"Files" ⇨ "Page Setup"命令，打开"Page Setup"对话框，在其中对打印结果进行页面设置，如图6.112所示。

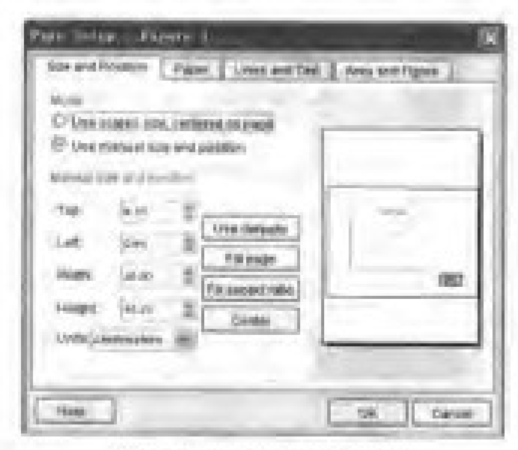


图6.112 设置打印页面

和其他常见软件（例如Word）打印页面类似，在MATLAB中打印图形之前，必须设置图形的属性，例如图形尺寸、纸张大小、线型以及文本类型等。在图6.112所示的对话框中，右方空白框中有一个图形框，这是一个示意图形框，可以如实地反映打印图形在纸面中的位置和

相对大小。用户可以直接使用鼠标左键来移动该图形框，如图 6.113 所示。

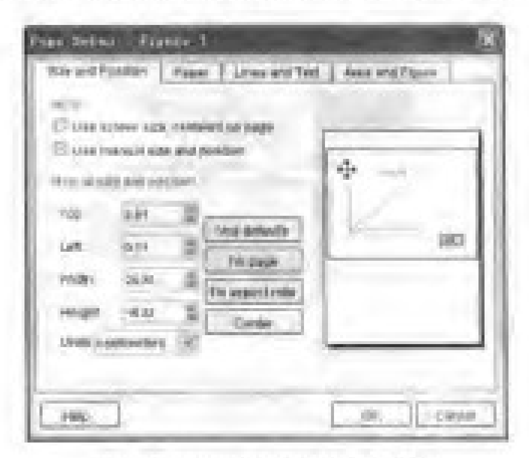


图 6.113　移动图形的位置

当用户将鼠标放置在示意图上方时，出现十字架的图标，可以自由移动该打印图形的位置，直到移到合适的位置为止。



可以在"Page Setup"对话框中的各个选项卡中设置图形的相应属性。限于篇幅，本小节中就不详细介绍各个选项卡的含义和作用，感兴趣的读者可以查阅相应的帮助文件。

step 2 预览待打印的图形。在上面的步骤中，已经设置了图形的属性，这就基本完成了打印前的基本设置工作。现在可以预览待打印的图形，选择图形窗口中的"Files"⇨"Print Preview"命令，打开"Print Preview"对话框，可以在其中预览待打印的图形，如图 6.114 所示。

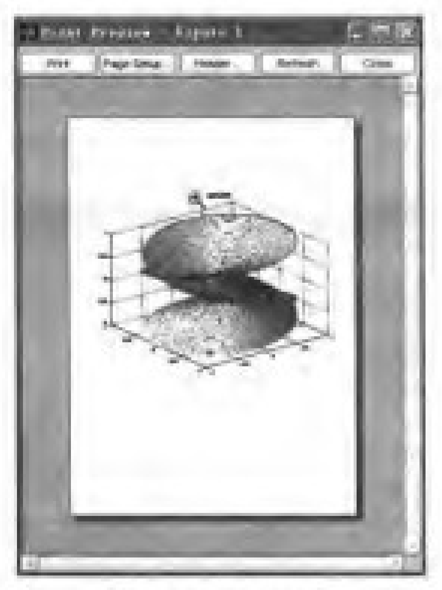


图 6.114　图形预览

当将鼠标放在图形的预览区时，鼠标出现自动缩放的符号；当单击鼠标左键时，可以放大局部的图形；当单击鼠标右键时，可以缩小局部的图形。

step 3 设置图形的标题。单击"Print Preview"对话框中的"Header..."按钮，打开"Figure Page

Header"对话框，可以在该对话框中设置图形的标题和日期格式，如图 6.115 所示。



图 6.115 设置图形的标题

单击"Figure Page Header"对话框中的"OK"按钮可以完成对图形标题的设置工作，设置的效果直接出现在完成的图形中，然后关闭该对话框，MATLAB 自动保存上面所有的设置工作。



在"Figure Page Header"对话框中，还可以设置不同字体的图形标题。单击上面对话框中的"Header Font"按钮，打开"字体"对话框，在该对话框中选择标题的字体格式和大小。

step 4 打印设置。关闭"Figure Page Header"对话框，返回图形窗口中。选择菜单栏中的"Files" ⇨ "Print Setup"命令，打开"打印设置"对话框，在其中设置关于打印机的属性，如图 6.116 所示。



图 6.116 设置打印机的属性

前面的步骤中，已经完成关于图形打印的相关图形设置，返回图形窗口中，选择菜单栏中的"Files" ⇨ "Print"命令，直接打印该 MATLAB 的图形。

6.10.2 图形打印的命令操作方式

在 MATLAB 中，关于图形打印的常见调用命令如下。

- print 将图形发送到由 printopt 定义的打印设置和系统打印命令中。
- print filename 将图形输出到文件 filename 中。如果 filename 没有扩展名，print 命令自动选择一个扩展名。
- print -ddriver 使用由 ddriver 定义的打印设置打印当前图形。
- print -dformat 将当前图形复制到系统粘贴板上。
- print -dformat filename 以用户自定义的图形格式将图形输出到用户自定义的 filename 文件中。
- print -smodelname 打印当前 simulink 模型 smodelname。
- print ... -options 定义打印选项。
- [pcmd,dev] = printopt 返回当前系统的打印命令到字符串变量 pcmd 和输出设备到变量 dev 中。

下面简单使用一个实例，说明如何在 MATLAB 中合理运用上面的命令。

例 6.66 使用 MATLAB 的命令操作方式，打印 $y=\sin t$ 函数的图形。

在 MATLAB 命令窗口中输入下面的代码：

```
>> t=(1:100)/100*4*pi;
>> y=sin(t);
>> plot(t,y);
>> title('Print Figure')
>> print
```

在输入上面的代码后，图形会在 Windows 打印程序的管理下通过默认的打印机输出。



小结

本章从各个方面介绍了在 MATLAB 中如何实现数据和函数的可视化，主要介绍了如何绘制二维和三维图形，以及如何设置图形外观的各种属性：颜色、光照、透明、材质等。熟练掌握本章中的常见命令，可以根据需要绘制各种个性化的图形。在后面的章节中，将介绍如何在 MATLAB 中进行程序设计。

◆ ◆ ◆

第7章　MATLAB7.0基础编程

本章包括

- M文件编辑器
- MATLAB的程序结构
- 向量化
- 程序的调试和剖析
- MATLAB的变量和关系式
- 控制语句
- 变量传递

MATLAB在工程运算方面有着广泛的应用，不仅有强大的数值运算功能、符号运算功能、矩阵运算和绘图功能，还可以像C语言、FORTRAN等高级语言一样进行程序设计，编写扩展名为.m的M文件。自行编写M文件可以灵活地解决各种实际问题，同时也可以在科学研究中加深应用。

MATLAB为用户编写M文件提供编辑器和编译器，用户可以使用这些工具完成复杂运算和应用。而且，MATLAB内置的函数大都是M文件函数，因此，了解M文件的结也有助于用户理解各种函数，也可以在原始的M文件基础上生成和扩展自己的函数库。

简单的来讲，所谓的M文件就是将处理问题的各种命令融合在一个文件中，该文件以.m为扩展名，然后由MATLAB系统进行编译，得出相应的运行结果，具有相当强大的可开发性和扩展性。同时，MATLAB的开发内核是C语言，具有简单易学的特点。学过C语言的读者会发现，MATLAB的许多语言规则都和C语言非常相似，用户可以很方便地在MATLAB开发环境中进行开发。

在本章中，将介绍MATLAB编程的各种基础知识，同时对于MATLAB7.0版本添加的新内容，例如函数句柄、程序性能优化等，也将加以介绍。

7.1　简单实例

在详细介绍MATLAB编程的知识之前，本节首先介绍一个比较简单的实例，介绍如何在MATLAB中编写函数文件和M文件。

本节主要介绍程序编写的操作步骤、编译过程等。对于各种语言结构和含义，将在后面章节中详细介绍。

7.1.1　编写函数文件

例7.1　在MATLAB中，编写对数据进行排序的函数文件。

step 1　打开MATLAB的文件编辑器。单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File"⇨"New"⇨"M-file"命令，打开M文件编辑器，如图7.1所示。

如果是第一次创建M文件，系统会将名字设置为"Untitled"（未命名）的。



M文件编辑器是用户编写、编译M文件的主要操作界面，本章中几乎所有的操作都是在该界面上完成的。关于该界面的菜单和按钮的功能，将在后面章节中介绍。



图 7.1　M 文件编辑器

step 2 在 M 文件编辑器中，编写 "ssort" 函数的代码，如图 7.2 所示。

图 7.2　编写 ssort 函数的脚本

上面函数程序的详细代码如下：

```
function out=ssort(a)
% SSORT 程序代码按照升序排列数据
% 需要提醒读者的是，该程序代码在排序方面没有太高的效率
% 如果需要对大型数据进行排序，请使用 MATLAB 的内置函数
% Define variables:
% a                 Input array to sort
% ii                Index variable
% iptr              Pointer to min value
% nvals             Number of values in "a"
% out               Sorted output array
% temp              Temp varible for swaping
nvals=size(a,2);
for ii=1:nvals-1
        iptr=ii;
        for jj=ii+1:nvals
              if a(jj)<a(iptr)
                     iptr=jj;
              end
        end
        if ii~=iptr
             temp=a(ii);
             a(ii)=a(iptr);
             a(iptr)=temp;
        end
```

```
end
out=a;
```

step 3 保存程序代码。当输入上面的代码后，单击文件编辑器工具栏中的保存图标，或者选择编辑栏中的"File" → "Save as"命令，打开"Save file as"对话框，在其中保存上面步骤编写的程序代码，如图 7.3 所示。



图 7.3 保存用户编写的程序代码

在默认情况下，MATLAB 将当前目录作为代码保存路径。在本例中，该目录就是"……\MATLAB7.0\work"。同时，由于在编写函数的时候，已经设置函数名称，在保存该代码的时候，就会将函数名称作为文件名，因此，本 M 文件的名称为 ssort.m。

在代码保存完成后，在 M 文件编辑器的标题栏中自动显示对应的保存路径。在本例中，得到的结果如图 7.4 所示。



图 7.4 保存后的程序代码



在前面的章节中，笔者曾经建议在使用MATLAB的时候，建立自己的程序目录，而不使用MATLAB的默认目录。在本实例中，为了简化步骤，采用了这种默认目录；但是，同样建议读者在保存代码时，选择自己创建的目录。

7.1.2 编写脚本文件

在MATLAB中，M文件编辑器除了可以编写函数文件之外，还可以编写脚本文件。在脚本文件中，除了没有定义函数名称之外，其他的和上面步骤类似，下面详细介绍。

例 7.2 在MATLAB中，编写对数据进行排序的脚本文件。

step 1 重新打开M文件编辑器，然后输入脚本文件的代码，如图7.5所示。



图7.5 输入脚本文件的代码

脚本文件的详细代码如下：

```
% Script file test_sort.m
%
% Purpose:
%   To read in an input data set,sort it into ascending order
% using the selection sort algorithm,and to write the sorted dat
% to the command window.
%  This program calls function "ssort" to do the actual sorting.
nvals=input('Enter number of the numbers to sort:');
array=zeros(1,nvals);
for ii=1:nvals
    string=[ 'Enter value ' int2str(ii) ':'];
    array(ii)=input(string);
end
    sorted=ssort(array);
    fprintf('\n sorted data:\n');
    for ii=1:nvals
        fprintf('%8.4f\n',sorted(ii));
    end
```

step 2 将上面的程序代码保存为"test_sort.m"文件，保存后的文件如图7.6所示。

7.1.3 运行代码

前面两个小节分别完成了函数代码和脚本代码，在本小节中，可以运行和检测代码。由于在test_sort的M文件中已经调用ssort函数，所以可直接运行test_sort文件，这样就将两个代码都检测了。下面详细介绍如何检测。



图 7.6 保存脚本文件代码

例 7.3 运行并检测前面小节编写的数字排序程序。

step 1 将保存程序的目录设置为 MATLAB 的当前目录，然后在 MATLAB 的命令窗口中输入 "test_sort"，按 "Enter" 键，得到的结果如图 7.7 所示。

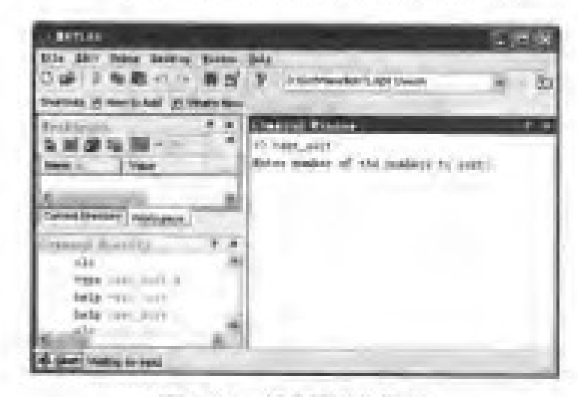


图 7.7 检测程序代码

可以看出，在命令窗口中输入脚本文件名称 "test_sort"，按 "Enter" 键后，MATLAB 会自动调用脚本程序语句的内容。同时，MATLAB 的命令窗口中出现 "Waiting for input"，表示系统处于接收数据的状态。

step 2 在程序的提示下，依次输入排序的数值，得到的结果如图 7.8 所示。

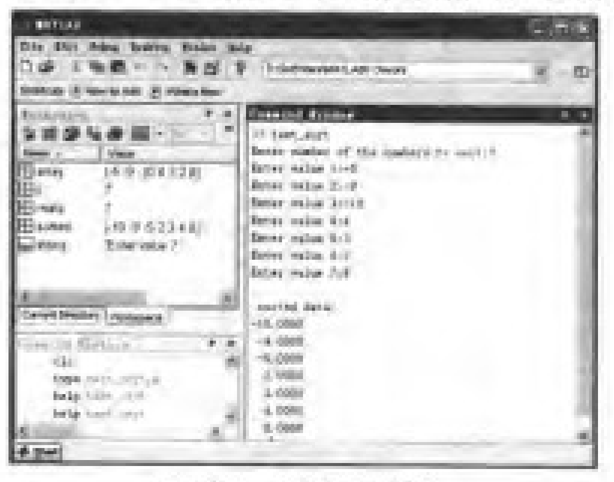


图 7.8 运行程序代码

当输入相应的数值后，MATLAB会调用代码中的相应代码，得出排序的结果；同时将程序处理过程中的变量填入工作空间中。

step 3 重新输入变量数值，查看新的排序结果，得到的结果如图7.9所示。

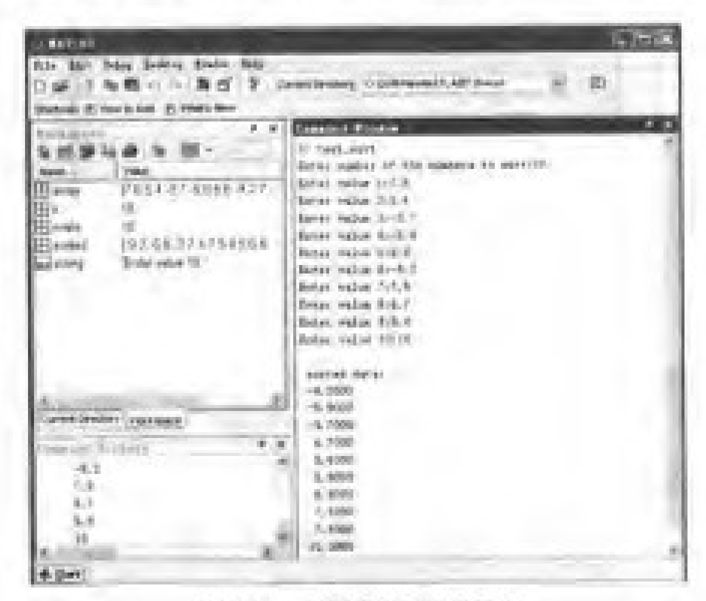


图7.9 重新运行程序代码

当在命令窗口中重新输入脚本文件的名称"test_sort"后，MATLAB再次调用相应的程序代码，可以在代码的提示下重新输入数值，得到新的排序结果。同时，MATLAB工作空间中的各个变量已经被覆盖成新的数值。



一般而言，函数程序代码过程中得到的中间变量都是局部变量，存放在函数自身的工作空间中；而脚本程序代码中得到的变量都是全局变量，存放在MATLAB的工作空间中。

7.1.4 检测代码

延续上面小节的步骤。

step 1 查看两段程序代码的帮助信息。在命令窗口中依次输入"help ssort"和"help test_sort"，查看代码的在线帮助信息，得到的结果如下：

```
>> help ssort
  SSORT Selection sort data in ascending order
  fuction ssort sorts a numeric data set into
  ascending order.Note that selection sort is
  relatively inefficient. When sorting large data
  sets,please use MATALAB's "sort" function.

>> help test_sort
  Script file test_sort.m
    Purpose:
    To read in an input data set,sort it into ascending order
  using the selection sort algorithm,and to write the sorted dat
  to the command window.
```

```
  This program calls function "ssort" to do the actual sorting.
```

关于该函数和脚本文件的帮助信息，就是前面步骤中用户在编写代码时，为各个代码添加的注释文字。对于ssort函数而言，注释文字不仅包括上面代码中显示的文字，之所以只显示该段文字，是因为在编写代码的时候，在后面一段注释文字前面添加了空格。



用户可以合理应用上面关于代码注释文字的规则，对于不希望显示的注释文字，可以在注释文字前面添加空格行。在本实例中，隐藏显示的注释文字是关于该段代码的变量定义，一般情况下不需要显示该帮助信息。

step 2 查看M文件的详细代码。在命令窗口中依次输入"type ssort"或"type ssort.m"，查看对应M文件的详细代码，得到的结果如下：

```
>> type ssort
function out=ssort(a)
% SSORT Selection sort data in ascending order
% fuction ssort sorts a numeric data set into
% ascending order.Note that selection sort is
% relatively inefficient. When sorting large data
% sets,please use MATALAB's "sort" function.

% Define variables:
% a               Input array to sort
% ii              Index variable
% iptr            Pointer to min value
% nvals           Number of values in "a"
% out             Sorted output array
% temp            Temp varible for swaping

nvals=size(a,2);
for ii=1:nvals-1
      iptr=ii;
      for jj=ii+1:nvals
          if a(jj)<a(iptr)
               iptr=jj;
          end
      end
      if ii~=iptr
          temp=a(ii);
          a(ii)=a(iptr);
          a(iptr)=temp;
      end
end
out=a;
```

7.2 M文件编辑器

上节已经使用M文件编辑器编写了基础的函数代码和脚本代码，本节将详细介绍该编辑器的使用方法和注意事项。

7.2.1 打开文件编辑器

M文件编辑器不会随着MATLAB的启动而启动，只有在用户编写M文件时，该编辑器才启动。需要提醒读者的是，M文件编辑器不仅可以用来编辑M文件，还可以对M文件进行交互性调试。而且，M文件编辑器还可以阅读和编辑其他ASCII码文件。通常情况下，可以使用下面的方法来打开M文件编辑器。

- 在上面介绍过，可以单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File" ⇨"New" ⇨ "M-file" 命令，打开M文件编辑器。这种方法适用于创建新的M文件。
- 如果需要编辑或者修改已经存在的M文件，可以单击MATLAB工具栏中的按钮，或者选择编辑栏中 "File" ⇨ "Open" 命令，打开Windows系统中标准的 "Open" 对话框，在对话框中选择需要编辑的M文件，然后单击对话框中的"打开"按钮，就可以打开该文件对应的编辑器。
- 除了在对话框中使用菜单选项打开M文件编辑器之外，还可以在命令窗口中输入 "edit" 命令，打开新的文件编辑器；或者输入 "edit filename" 打开一个存在M文件的编辑器，其中filename是M文件的名称，可以带扩展名也可以不带扩展名。



尽管在命令窗口中打开M文件编辑器比较简单，但是也有局限性。当打开的M文件没有保存在MATLAB的当前目录中时，系统会返回错误信息，无法打开对应的文件编辑器。

例7.4 使用窗口命令打开前面创建的ssort文件。

将"……\MATLAB7.0\work"目录设置为MATLAB的当前目录，在命令窗口中输入">> edit ssort"命令，然后按"Enter"键，得到的图形如图7.10所示。



图7.10 打开ssort文件

7.2.2 设置M文件编辑器

有过编程经历的人员也许都有这样的经验，不同的编程人员对编辑器的界面、字体、段落格式等都有自己的喜好。对此，MATLAB为用户提供了自定义这些属性的功能，用户可以选择相应的菜单选项来设置各种属性。

在本小节中，主要介绍设置M文件编辑器中的重要属性，同时还将介绍如何设置M文件的打印效果。之所以介绍如何设置M文件的打印效果，是因为在编写M文件程序的时候，用户经常需要编写相应的程序代码文件。

下面以实例为单位，介绍如何进行编辑器的设置工作。

例 7.5 自定义设置 M 文件编辑器的属性。

step 1 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的 "File" ⇨ "New" ⇨ "M-file" 命令，打开一个空白的M文件编辑器。然后选择文本编辑器菜单栏中的"File" ⇨"Preferences"命令，打开 "Preferences" 对话框，选中其中的 "colors" 选项，设置 M 文件编辑器中的各种字体颜色，如图 7.11 所示。



图 7.11　设置 M 文件的各类字体颜色

在上面对话框的左侧，可以为程序语句中的关键字（Keywords）、注释（Comments）、字符串（Strings）、未结束的字符串（Unterminated strings）、系统命令（System commands）和错误信息（Errors）等设置高亮显示的字体颜色。可以根据自己的喜好来设置各种字体颜色，只需要在对应选项右侧的选项中选择对应的颜色。

step 2 设置 M 文件编辑器中的显示选项。选择 "Preferences" 对话框中的 "Editor/Debugger" 选项下面的 "Display" 选项，选中 "Enable datatips in edit mode" 选项，如图 7.12 所示。



图 7.12　设置 M 文件编辑器的显示选项

在默认情况下，MATLAB 不会选中 "Enable datatips in edit mode" 选项，当选择该选项

后，将光标移动到编辑器文件中某变量名称时，就会出现一个现场菜单，显示该变量所存放的具体数据，这种设置有利于用户阅读程序代码。

step 3 设置M文件的保存选项。选择"Preferences"对话框中的"Autosave"选项，设置M文件的保存选项，如图7.13所示。



图7.13 设置M文件的保存选项

在上面的对话框中，可以设置M文件备份的选项，默认情况下，系统设置每5分钟保存用户编写的程序代码，用户可以根据自己的情况来修改这个数值。



完成上面的所有设置工作后，单击"Apply"按钮，就可以将自定义的设置应用在后面编写的程序代码中。设置完成所有的选项属性后，单击"OK"按钮，就可以退出该对话框。

7.2.3 设置M文件编辑器的打印属性

例7.6 自定义设置M文件编辑器的打印效果属性。

step 1 在上面实例中打开的M文件编辑器中，选择菜单栏的"File" ⇨ "Page setup"命令，打开"Page Setup: Editor"对话框，选中"Layout"选项卡，设置M文件的版面布局，如图7.14所示。



图7.14 设置M文件的版面布局

在默认情况下，MATLAB勾选该对话框中的"Print header"选项，则打印的页面中会显示M文件所在的完整路径、文件创建日期和页数等信息。如果是比较长的程序代码，可以勾选"Print line numbers"选项，则会在程序代码中出现代码的行数标记。

step 2 设置版头属性。选择"Page Setup: Editor"对话框中的"Header"选项卡，设置M文件的版头属性，如图7.15所示。



图7.15 设置版头属性

在上面的对话框中可以设置版头的边框、页数的显示方式和布局等属性。在本实例中，选择"Simple two line"的布局，将版头信息分成两行显示；选择"shaded box"的边框属性，设置带有阴影的边框。

step 3 设置打印的字体属性。选择"Page Setup: Editor"对话框中的"Fonts"选项卡，设置M文件的打印字体属性，如图7.16所示。



图7.16 设置打印的字体属性

在默认情况下，MATLAB会选中"Use Editor font"选项，这样打印代码出现的结果就会和M文件编辑器中的字体相同。如果希望使用自行设置的字体，可以首先选择"Use custom font"选项，然后在出现的下拉菜单中选择相应的字体信息。

7.3 MATLAB的变量和关系式

在MATLAB中，可以进行常用的加、减、乘、除和幂等运算。对于简单的计算可以直接在命令窗口中输入表达式，MATLAB会将计算的结果保存在默认的ans变量中。与C语言不同，在MATLAB中使用变量可以不预先定义。

尽管MATLAB的程序内核是C语言，但是在具体的变量和语法规则上，MATLAB还是和C语言有着不同的要求和体系。

7.3.1 M 文件的变量类型

在复杂的程序结构中，变量是各种程序结构的基础。因此，MATLAB 中的变量也有自己的命令规则：必须以字母开头，之后可以是任意字母、数字或者下划线，同时变量命名不能有空格，变量名称区分大小写。最后，在 MATLAB7.0 中，变量名称不能超过 63 个字符，第 63 个字符之后的部分都将被忽略。

在MATLAB中有一些默认的预定义变量，用户在设置变量时应该尽量避免和这些默认的变量不同，否则会给程序代码带来不可预测的错误，表 7.1 列出了常见的预定义变量。

表 7.1　MATLAB 中的预定义变量

预定义变量	含义
ans	计算结果的默认名称
eps	计算机的零阈值
inf(Inf)	无穷大
pi	圆周率
NaN(nan)	表示结果或者变量不是数值

在编写程序代码的时候，可以定义全局变量和局部变量两种类型，这两种变量类型在程序设计中有着不同的应用范围和工作原理。因此，有必要了解这两种变量的使用方法和特点。

当每一个函数在运行的时候，都会占有独自的内存，这个工作空间独立于 MATLAB 的基本工作空间和其他函数的工作空间。这样的工作原理保证了不同的工作空间中的变量相互独立，不会相互影响，这些变量都被称为局部变量。

在默认情况下，如果用户没有特别声明，函数运行过程中使用的变量都是局部变量。如果希望减少变量传递，可以使用全局变量。在 MATLAB 中，定义全局变量需要使用命令 global，其调用格式如下：

```
global Var1 Var2
```

通过上面简单的命令，就可以使MATLAB允许几个不同的函数空间以及基本工作空间共享同一个变量。每个希望共享全局变量的函数或者MATLAB基本工作空间必须逐个对具体变量进行专门的定义。如果某个函数在运行过程中修改了全局变量的数值，则其他函数空间以及基本工作空间内的同名变量数值也会随之变化。



尽管MATLAB对全局变量的名称没有特别的限制，但是为了提高程序的可读性，笔者建议采用大写字符来命令全局变量。同时，将全局变量尽量放在函数体的首位。

7.3.2 M 文件的关键字

在命令变量名称时，MATLAB 预留了一些关键字并且不允许用户对其进行重载。因此在定义变量名称的时候，应该避免使用这些关键字，否则系统会显示类似于缺少操作数之类的错误提示。在 MATLAB 中，可以使用 "iskeyword" 命令来查看 MATLAB 中的关键字，得到的结果如下：

```
>> iskeyword
ans =
    'break'
```

```
'case'
'catch'
'continue'
'else'
'elseif'
'end'
'for'
'function'
'global'
'if'
'otherwise'
'persistent'
'return'
'switch'
'try'
'while'
```

关系表达式

在MATLAB的常见分支或者循环控制结构中，经常会遇到判断结构，根据某种条件的数值0或者1而得出不同的结论，因此首先需要通过某种表达式来产生这种逻辑上的判断数值0或者1。在MATLAB中，能够产生这种逻辑数值0或者1的表达式有关系表达式和逻辑表达式。在本小节中，将详细介绍如何使用关系表达式。

关系表达式是针对两个变量的表达式，可能是两个数值变量或者字符串变量，通过表达式之间的关系得出逻辑值0（false）或者1（true），取决于两个变量之间的关系。

关系表达式的通用命令如下：

```
a op b
```

其中，a和b可以是算术表达式、变量、字符串等，op是一种逻辑关系。如果上面的表达式表达的关系是正确（true）的，则表达式返回数值1；如果表达式表达的关系是错误的，则返回数值0。

在表7.2中列出了MATLAB中常见的逻辑关系。

表7.2 MATLAB中的常见逻辑关系

关系运算符	含义
==	相等
~=	不等
>	大于
>=	大于等于
<	小于
<=	小于等于

例7.7 在MATLAB中，使用关系运算符进行运算，得到相应的结果。

step 1 在MATLAB的命令窗口中输入下列内容：

```
>> op1=(3<4);
>> op2=(3<=4);
>> op3=(3==4);
>> op4=(3>4);
>> op5=(4<=4);
>> op6=('A'<'B');
```

```
>> op=[op1;op2;op3;op4;op5;op6];
```

step 2 查看上面的关系表达式中，op 数组储存的就是各个表达式的结果，查看其数值，得到的结果如下：

```
>> op
op =
     1
     1
     0
     0
     1
     1
```

step 3 查看逻辑运算数值的变量类型，得到的结果如下：

```
>> class(op)
ans =
logical
```



在上面的6个关系表达式中，最后一个表达式比较的是字符串，之所以返回的数值是1，是因为在MATLAB中字符串关系是按照字母表次序来排列的。最后，通过class函数来判断op数组的变量类型，得到的结果就是logical类型。

例 7.8 在 MATLAB 中，以数值矩阵（数组）为单位，进行关系表达式的运算。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> a=[-1,0;-2,1];
>>b=0;
>>c=[0,2;-2,-1];
>>d=[0,2,3;1,3,5];
>>Op1=(a>b);
>>Op2=(a>c);
```

step 2 查看上面关系表达式的运算结果如下：

```
>> Op1
Op1 =
     0     0
     0     1
>> Op2
Op2 =
     0     0
     0     1
```

step 3 将上面的数值 c 和 d 进行关系运算，得到的结果如下：

```
>> Op3=(c>d)
??? Error using ==> gt
Matrix dimensions must agree.
```

从上面的程序代码中可以看出，一个数值矩阵或者数组可以和一个标量进行关系运算。其运算规则是将矩阵的数值依次和标量数值进行关系运算，得出相应的关系结果，返回一个逻辑的矩阵。同时，同维的矩阵也可以相互进行关系运算，运算规则是将对应数值进行关系运算，同样可以得到一个逻辑矩阵。

如果将不同维的矩阵进行关系运算，MATLAB 将无法完成其关系运算，返回相应的错误信息，提示用户两个矩阵应该同维。



在MATLAB中，字符串变量本身就是一个矩阵变量，因此关系表达式只能比较长度相同的字符串变量，否则 MATLAB 就会返回错误信息。

7.3.4 关系表达式的优先级

例 7.9 在 MATLAB 中分析关系表达式和算术运算的优先级。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> Op1=4+5<2;
>> Op2=(4+5)<2;
>> S1=(Op1==Op2);
>> Op3=6-2>5;
>> Op4=(6-2)>5;
>> S2=(Op3==Op4);
```

step 2 查看上面关系运算的结果，得到的结果如下：

```
>> A=[Op1,Op2,S1]
A =
     0     0     1
>> B=[Op3,Op4,S2]
B =
     0     0     1
```

通过上面的程序代码可以看出，无论用户是否使用括号来进行算术运算，得到的结果都是相等的。也就是说，上面的 Op1 和 Op2 是完全相同的。这就表明在 MATLAB 运算中，算术表达式的优先级高于关系表达式，是否添加括号不会影响最后的关系运算结果。

关于下面的表达式 Op3 和 Op4，原理和上面分析的完全相同。由于加、减是算术运算的最底层，因此本实例使用加、减运算和关系运算进行比较。



尽管在MATLAB中，算术表达式的优先级高于关系表达式，在进行关系运算时可以不添加括号，就可以得出正确的结果。但是，根据笔者的经验，当进行复杂的算术表达式的关系运算时，还是建议使用相应的括号，避免产生运算中的误解。

7.3.5 截断误差问题

例 7.10 在 MATLAB 中对数值进行关系运算。

step 1 在 MATLAB 的命令窗口中输入下列内容:

```
>> a=0;
>> b=cos(3*pi/2);
>> op1=(a==b);
>> op2=(a~=b);
```

step 2 查看上面关系运算的结果如下:

```
>> op1
op1 =
     0
>> op2
op2 =
     1
```

根据上面程序代码的结果可以看出,尽管$b=\cos(\frac{3}{2}\pi)=0$,数值a也等于0,但是MATLAB却认为两个变量数值不相等,因此op1的逻辑数值为0,op2的逻辑数值为1。造成这样结果的原因是,MATLAB在数值运算中的截断误差(roundoff error)。在MATLAB中,表达式$\cos(\frac{3}{2}\pi)$得到的数值并不直接等于0,而是等于1.2246×10^{-16},所以,当将变量a和b进行比较时,返回就是逻辑值0(false)。

step 3 修改程序代码,重新比较两个数值大小:

```
>> op3=abs(a-b)<1.0e-15
op3 =
     1
```

在上面的程序代码中,为了避免MATLAB中的数值运算截断误差,将两个数值a和b进行比较时,使用的表达式为$|a-b|<1.0\times10^{-15}$,也就是说,判断两个数值变量之间的数值间隔是否足够小,这个数值就可以避免MATLAB中的运算截断误差。

逻辑表达式

在MATLAB中,逻辑表达式是通过逻辑关系符将两个或者多个进行连接得到的逻辑值。在MATLAB中,提供三种比较常见的逻辑运算符:AND、OR和XOR。这三种逻辑运算符是二元关系运算符,同时,MATLAB还提供了一个非二元关系运算符NOT。

在表7.3中列出了MATLAB中常见的逻辑运算符。

表7.3 MATLAB中的常见逻辑运算符

关系运算符	含义
&	逻辑 AND
|	逻辑 OR
Xor	逻辑 Xor
~	逻辑 NOT

例7.11 在MATLAB中,使用逻辑表达式进行数值逻辑运算。

step 1 在MATLAB的命令窗口中输入下列内容:

```
>> value1=1;
>> value2=0;
>> value3=-10;
>> op1=~value1;
>> op2=value1|value2;
>> op3=value1&value2;
>> op4=value1&value2|value3;
>> op5=value1&(value2|value3);
>> op6=~(value1&value3);
>> op=[op1,op2,op3,op4,op5,op6];
```

step 2 查看逻辑运算的结果如下：

```
>> op

op =
     0     1     0     1     1     0
```

例 7.12 在 MATLAB 中，综合使用逻辑和关系运算得出逻辑结果。

step 1 在 MATLAB 的命令窗口中输入下列内容：

```
>> a=2;
>>b=[-1,-2;0,10];
>>op1=~(a>b);
>>op2=(~a)>b;
>>op3=~a>b;
>> op=(op2==op3);
```

step 2 查看上面的运算结果如下：

```
>> op1
op1 =
     0     0

>> op2
op2 =
     1     1
     0     0
>> op3
op3 =
     1     1
     0     0
>> op
op =
     1     1
     1     1
```

上面的实例并不复杂，主要是为了介绍关系表达式和逻辑表达式的优先级问题。由于表达式中 op2 和 op3 是完全相同的，因此可以看出逻辑运算符 ~ 的优先级要高于关系运算符>。下面简要介绍 MATLAB 中关系运算符和逻辑运算符的优先级情况，包括算术运算符在内，其优先级依次为：

- 所有的算术运算符的优先级最高，高于关系和逻辑运算符；
- 其次是所有的关系运算符（==，~=,>,>=,<,<=），关系运算符从左向右运算；

- 接着是逻辑上的 NOT 运算符 ~；
- 接着是逻辑上的 AND 运算符 &，运算顺序是从左向右；
- 最后是逻辑上的 XOR 运算符 |，运算顺序是从左向右。



逻辑运算函数

在本小节中，将向读者介绍MATLAB中的逻辑运算函数，或者称为 is类函数。当函数判断的条件成立时，该类函数将返回逻辑数值 1；当函数判断的条件不成立时，该类函数将返回逻辑数值 0。这些函数都可以用在关系表达式或者逻辑表达式中，也可以用来作为程序结构的判断条件。

表 7.4 列出了 MATLAB 中常见的逻辑函数。

表 7.4　MATLAB 中的常见逻辑函数

函数名称	功能
ischar	判断变量是否是字符串变量
isempty	判断变量是否是空白数组
isinf	判断数值变量是否是无穷大
isnumeric	判断变量是否是数值数组



MATLAB 的程序结构

和其他编程语言类似，MATLAB 也给用户提供判断结构来控制程序流的执行次序。一般来讲，决定程序结构的语句有顺序结构、分支结构和循环结构三种，每种语句结构都有各自的流控制机制，相互配合使用可以实现功能强大的程序。

由于 MATLAB 的这种控制命令用法和其他语言用法十分类似，因此本节只结合 MATLAB 的特点对控制命令进行简要的介绍。



顺序结构

顺序结构是最遵循逻辑思路的程序代码结构，批处理文件就是典型的顺序语句的文件，这种语句不需要任何特殊的流控制。这种顺序结构是最基础的程序结构，也是其他控制流语句中的重要组成部分。

例 7.13　在 MATLAB 中，使用顺序结构编写绘制函数的图形。

step 1 单击 MATLAB 命令窗口工具栏中的 □ 按钮，打开 M 文件编辑器。在 M 文件编辑器中输入下面的程序代码：

```
%定义符号变量 t 和 tao
syms t tao
%定义积分表达式
y=exp(-t/3)*cos(1/2*t);
%对表达式进行积分
s=subs(int(y,0,tao),tao,t);
%绘制积分图形
ezplot(s,[ 0,4*pi] );grid
```

step 1 单击 M 文件编辑器中的“保存”按钮，将上面的程序代码保存为“ezexm”，得到的结果如

图7.17所示。



图7.17 保存M文件程序代码

step 3 返回到MATLAB的命令窗口，输入"ezexm"，然后按"Enter"键，得到的结果如图7.18所示。

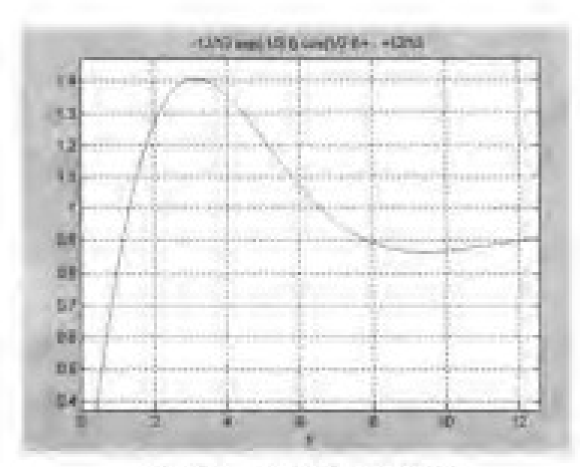


图7.18 得到的程序结果

在上面的程序代码中，首先定义符号变量，然后定义积分表达式，进行积分运算，最后调用ezplot命令绘制积分函数的图形。这样的程序代码流程符合逻辑顺序，而且容易阅读，容易理解，这是顺序结构的重要优点。



在上面的程序代码中，使用了ezplot的绘图命令，对该命令的详细使用方法感兴趣的读者可以阅读本书的相关章节。

7.4.2 if分支结构

如果在程序中需要根据一定条件来执行不同的操作时，可以使用条件语句，在MATLAB中提供if分支结构，或者称为是if-else-end语句。

根据不同的条件情况，if分支结构有多种形式，其中最简单的用法是：如果条件表达式expression为真，则执行组命令statements，否则跳过该组命令，其格式如下：

```
if expression
   Statements
end
```

如果上面的expression是一个空数组，MATLAB则认为条件表达式为假。在MATLAB中，if分支

结构的更为普遍的调用格式如下：

```
if expression1
    statements1
elseif expression2
    statements2
.....................
else
    statementsk
end
```

如果条件语句只有两种选择的可能，则其调用格式如下：

```
if expression1
    statements1
else expression2
    statements2
end
```

在大多数情况下，条件表达式会由关系表达式或者逻辑表达式组成，这些表达式返回的都是逻辑值 0 或者 1，将作为条件判断的依据。为了提高程序代码执行的效率，MATLAB 会尽可能少地检测这些表达式的数值。

例 7.14 在 MATLAB 中，使用 if 分支结构编写求解一元二次方程 $ax^2+bx+c=0$ 的程序代码，并且运行检测该代码结果。

step 1 分析分支结构的判断条件。根据基础数学知识，一元二次方程 $ax^2+bx+c=0$ 的根的性质直接取决于判别式 $\Delta=b^2-4ac$ 的数值。当 $\Delta=0$ 时，该方程有两个相等的实根；当 $\Delta>0$ 时，该方程有两个互不相等的实根；当 $\Delta<0$ 时，该方程有两个虚根。

step 2 单击 MATLAB 命令窗口工具栏中的 D 按钮，打开 M 文件编辑器。在 M 文件编辑器中输入下面的程序代码：

```
% script file calc_root.m
%
% purpose:
% This program solves for the roots of a quadratic equation
% of the form a*x^2+b*x+c=0.It calcaulates the answers of
% roots the equation posseses.
%
% Define variables:
% a        coefficient of x^2
% b        coefficient of x
% c        constant term
% x1       first root of the equation
% x2       second root of the equation

disp('This program solves for the roots of a quadratic equation');
disp('of the form a*x^2+b*x+c=0');
a=input('Enter the coefficient A:');
b=input('Enter the coefficient B:');
c=input('Enter the coefficient C:');
discriminant=b^2-4*a*c;

% 如果判别式大于 0
```

```
%则根据二元方程的公式得出两个不同的实数解
if discriminant>0
    x1=(-b+sqrt(discriminant))/(2*a);
x2=(-b-sqrt(discriminant))/(2*a);
%在命令窗口显示求解结果
    disp('This equation has two real roots');
    fprintf('x1=%f\n',x1);
fprintf('x2=%f\n',x2);

%当判别式等于 0，则返回两个相同的实数根
elseif discriminant==0
    x1=-b/(2*a);
    disp('This equation has two identical roots');
    fprintf('x1=x2=%f\n',x1);

%当判别式小于 0，则返回两个虚根
else
    real_part=-b/(2*a);
    image_part=sqrt(abs(discriminant))/(2*a);
    disp('This equation has two complex roots');
    fprintf('x1=%f+i%f\n',real_part,image_part);
    fprintf('x2=%f-i%f\n',real_part,image_part);
end
```

step 3 单击 M 文件编辑器中的“保存”按钮，将上面的程序代码保存为“calc_root.m”，得到的结果如图 7.19 所示。



图 7.19 保存程序代码

step 4 返回到 MATLAB 的命令窗口，输入“calc_root”，然后按“Enter”键，根据程序代码的提示，依次输入方程的系数，得到的结果如下：

```
>> calc_root
This program solves for the roots of a quadratic equation
of the form a*x^2+b*x+c=0
Enter the coefficient A:1
Enter the coefficient B:5
Enter the coefficient C:6
This equation has two real roots
x1=-2.000000
x2=-3.000000
```

```
>> calc_root
This program solves for the roots of a quadratic equation
of the form a*x^2+b*x+c=0
Enter the coefficient A:1
Enter the coefficient B:4
Enter the coefficient C:4
This equation has two identical roots
x1=x2=-2.000000
>> calc_root
This program solves for the roots of a quadratic equation
of the form a*x^2+b*x+c=0
Enter the coefficient A:1
Enter the coefficient B:2
Enter the coefficient C:5
This equation has two complex roots
x1=-1.000000+i2.000000
x2=-1.000000-i2.000000
```



在上面的程序中，分别输入不同情况下的系数数值，得出不同条件下方程的解。这样就可以检测程序代码中 if 分支结构的语句程序的有效性。

最后，在使用 if 分支结构时，需要注意下面的两个问题。

- if 分支结构是所有程序结构中最灵活的结构之一，可以使用任意多个 esleif 语句，但是只能有一个 if 语句和一个 end 语句。
- if 语句可以相互嵌套，可以根据实际需要将各个 if 语句进行嵌套，从而解决比较复杂的实际问题。

7.4.3 switch 分支结构

和C语言中的switch分支结构类似，在MATLAB中适用于条件多而且比较单一的情况，类似于一个数控的多个开关。其一般的语法调用方式如下：

```
switch expression (scalar or string)
   case value1
      statements
   case value2
      statements
   ......................
   otherwise
      statements
end
```

在上面的语法结构中，expression 是一个标量或者字符串，MATLAB 可以将表达式中的数值依次和各个 case 命令后的数值进行比较。如果比较结果为假，MATLAB 会取下一个数值来进行比较，一旦比较结果为真，MATLAB 执行相应的命令，然后跳出该分支结构。如果所有的比较结果都为假，也就是表达式的数值和所有的检测值都不相等，MATLAB 执行 otherwise 部分的语句。

例 7.15 在 MATLAB 中，使用 switch 分支结构来判断用户输入的数值。

step 1 单击 MATLAB 命令窗口工具栏中的□按钮，打开 M 文件编辑器。在 M 文件编辑器中输入下面的程序代码：

```
%提示用户输入数值
input_num=input('Enter the number:');
%根据情况判断数值大小，显示数值信息
switch input_num
    case -1
        disp('negative one');
    case 0
        disp('zero');
    case 1
        disp('positive one');
        %如果不是上面的数值，显示'其他数值'
    otherwise
         disp('other value');
end
```

step 2 单击M文件编辑器中的"保存"按钮，将上面的程序代码保存为"calc_root.m"，得到的结果如图7.20所示。



图 7.20 保存程序代码

step 3 返回MATLAB的命令窗口，输入"swcase"，然后按"Enter"键，根据程序代码的提示，依次输入不同的数值，得到的结果如下：

```
>> swcase
Enter the number:1
positive one
>> swcase
Enter the number:-1
negative one
>> swcase
Enter the number:0
zero
>> swcase
Enter the number:8
other value
```



在switch分支结构中，case命令后的检测不仅可以为一个标量或者字符串，还可以为一个元胞数组。如果检测值是一个元胞数组，MATLAB将把表达式的值和该元胞数组中的所有元素进行比较；如果元胞数组中某个元素和表达式的值相等，MATLAB认为比较结果为真。

7.4.4 try–catch 结构

在MATLAB中，try–catch结构的功能和error类似，主要用来对异常情况进行处理。其相应的语法结构如下：

```
try
   statement
   ...
   statement
catch
   statement
   ...
   statement
end
```

在上面的语法结构中，try后面的命令语句会被执行，只有当这些语句执行出现错误时，catch控制语句就会捕获它，执行相应的语句。如果执行catch语句后的命令又出现错误，MATLAB就会终止该程序结构。

例 7.16 在MATLAB中，使用try–catch语法结构判断数组错误。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
% 初始化数组
a=[ 1 -3 2 5];
try
% 显示某个元素
   index=input('Enter subscript of element to display: ');
   disp([ 'a(' int2str(index) ') = ' num2str(a(index))] );
catch
  % 如果发现出错
  disp([ 'Illegal subscript: ' int2str(index)] );
  % 显示错误类型
  A=lasterr;
  disp([ 'The type of error: ' A ]);
end
```

step 2 单击M文件编辑器中的"保存"按钮，将上面的程序代码保存为"calc_root.m"，得到的结果如图7.21所示。

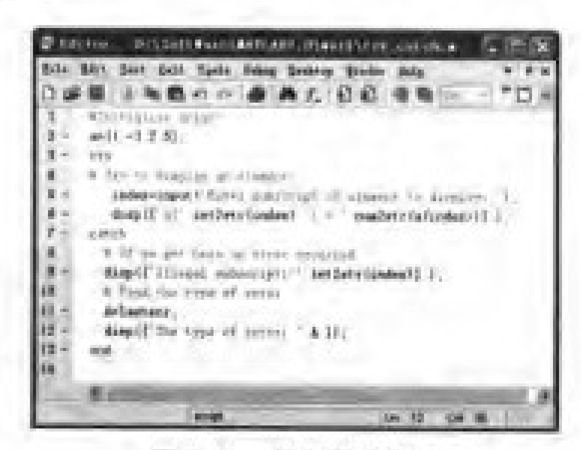


图 7.21 保存程序代码

step 3 返回MATLAB的命令窗口，输入"try_catch"，然后按下"Enter"键，根据程序代码的提示，依据不同的数值情况，得到的结果如下：

```
>> try_catch
Enter subscript of element to display: 2
a(2) = -3
>> try_catch
Enter subscript of element to display: 6
Illegal subscript: 6
The type of error: Index exceeds matrix dimensions.
```

从上面的实例可以看出，当输入的数组序号超过了数组的维度时，MATLAB会显示错误信息，同时显示系统的错误提示"Index exceeds matrix dimensions"。



在MATLAB中，lasterr命令的功能是显示最后的错误信息，该命令返回的信息是字符串类型。它经常和try-catch结构配合起来使用，显示错误的信息。关于该命令的详细用法，可以查阅MATLAB中的帮助文件。

7.4.5 while 循环结构

除了上面介绍的分支结构之外，MATLAB还提供多个循环结构。和其他编程语言类似，循环语句一般用于有规律的重复计算。被重复执行的语句称为循环体，控制循环语句流程的语句称为循环条件。

在MATLAB中，while循环结构的语法形式如下：

```
while expression
    statements
end
```

while循环的次数是不固定的，只要表达式的值为真，循环体就会被执行。通常表达式给出的是一个标量，但是也可以是数组或者矩阵；如果是后者，那么while循环要求所有的元素数值为真。

例 7.17 在MATLAB中，使用while循环结构对数据进行统计，计算输入数值数组的平均值和标准差。

step 1 分析程序的数学公式。根据统计学的基础知识，可以分析得到样本的平均值和标准差的公式分别如下：

$$\bar{x}=\frac{1}{N}\sum_{i=1}^{N}x_i$$

$$s=\sqrt{\frac{N\sum_{i=1}^{N}x_i^2-(\sum_{i=1}^{N}x_i)^2}{N(N-1)}}$$

其中，参数N表示用户输入的数值个数，x_i表示各个输入的数值。对于这两种基础统计量，MATLAB提供mean和std命令来计算。在本实例中，为了演示while循环结构，编写计算这些参量的简单代码。

step 2 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面

的程序代码:

```
% Script file stats.m
%
% Purpose:
%   To calculate mean and the standard deviation of
% an input data set containing and arbitrary number
% of input values.
%
% Define variables:
%  n        The number of input samples
% std_dev   The standard devation of the input samples
% sum1      The sum of the input values
% sum2      The sum of the squares of the input values
% x         input data value
% xvar      The average of the input samples
% Initalize variables
n=0;sum1=0;sum2=0;
%Read the input values
x=input('Enter the first value: ');

% 创建求解数组元素和的循环结构
while x>=0
    n=n+1;
    sum1=sum1+x;
    sum2=sum2+x^2;

    % 读入原始数据
    x=input('Enter next value: ');
end
% 计算平均值和标准方差
xvar=sum1/n;
std_dev=sqrt((n*sum2-sum1^2)/(n*(n-1)));
% 在命令窗口显示结果
fprintf('The mean of this data set is: %f\n',xvar);
fprintf('The standard deviation is: %f\n',std_dev);
fprintf('The number of data is: %d\n',n);
```

step 3 单击M文件编辑器中的“保存”按钮，将上面的程序代码保存为“stats.m”，得到的结果如图7.22所示。

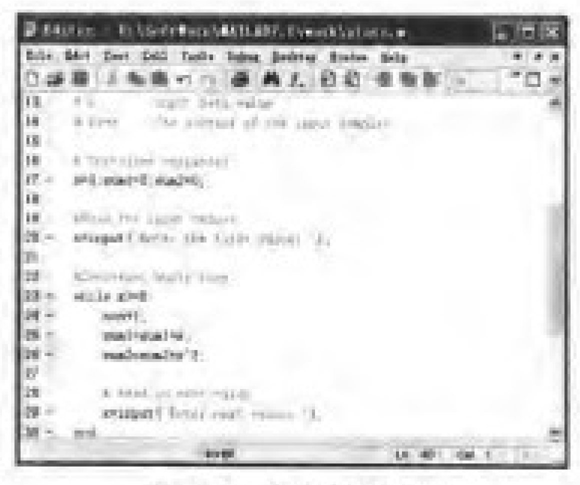


图7.22　保存程序代码

step 4 返回MATLAB的命令窗口，输入“stats”，然后按“Enter”键，根据程序代码的提示，依据不同的数值情况，得到的结果如下：

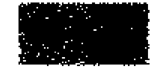
```
>> stats
Enter the first value: 6
Enter next value: 7
Enter next value: 9
Enter next value: 12
Enter next value: 0
Enter next value: 5
Enter next value: -7
The mean of this data set is: 6.500000
The standard deviation is: 4.037326
The number of data is: 6
```

从上面的实例可以看出，当输入任意的负数时，MATLAB会自动跳出while循环结构，只统计前面输入非负数值的统计量，得到相应的统计结果。

step 5 使用MATLAB的内置函数进行数值统计，检测程序是否正确，得到的结果如下：

```
>> A=[ 6,7,9,12,0,5] ;
>>mean_A=mean(A,2);
>>std_A=std(A,0,2);
>> size_A=size(A,2);
>> mean_A
mean_A =
    6.5000
>> std_A
std_A =
    4.0373
>> size_A
size_A =
     6
```

从上面的结果中可以看出，使用MATLAB内置的函数进行数据量的统计时，得到的结果和上面编写的程序结果相同。

 for循环结构

在MATLAB中，另外一种常见的循环结构是for循环结构，其常用的调用格式如下：

```
for variable = expression
    statements
end
```

在上面的语法结构中，variable称为循环变量，循环体被重复执行的次数是确定的，该次数由for命令后的表达式expression决定。

例7.18 在MATLAB中使用for循环结构完成上面小节的数值统计功能。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
% Script file stats2.m
```

```
%
% Purpose:
%   To calculate mean and the standard deviation of
%  an input data set containing and arbitrary number
%  of input values.
%
% Define variables:
%  n        The number of input samples
% std_dev    The standard devation of the input samples
% sum1      The sum of the input values
% sum2      The sum of the squares of the input values
% x         input data value
% xvar      The average of the input samples
% Initalize variables
sum1=0;sum2=0;

% 输入排序数据的个数
n=input('Enter the number of points: ');
% Check to see if we have enough input data.
if n<2
    disp('At least 2 values must be entered.');
else
     % 创建计算总和和平方和的循环
     for ii=1:n
         x=input('Enter value: ');
         sum1=sum1+x;
         sum2=sum2+x^2;
      end
%计算平均值和标准方差
xvar=sum1/n;
std_dev=sqrt((n*sum2-sum1^2)/(n*(n-1)));
%Print the result
fprintf('The mean of this data set is: %f\n',xvar);
fprintf('The standard deviation is: %f\n',std_dev);
fprintf('The number of data is: %d\n',n);
end
```

step 2 单击 M 文件编辑器中的"保存"按钮，将上面的程序代码保存为"stats2.m"，得到的结果如图 7.23 所示。



图 7.23　保存程序代码

step 3 返回到MATLAB的命令窗口，输入"stats2"，然后按"Enter"键，根据程序代码的提示，依据不同的数值情况，得到的结果如下：

```
>> stats2
Enter the number of points: 7
Enter value: 5
Enter value: 6
Enter value: -7
Enter value: 0
Enter value: -2
Enter value: 3
Enter value: 4
The mean of this data set is: 1.285714
The standard deviation is: 4.608481
The number of data is: 7
```

在上面的程序代码中，使用for循环替代了前面小节中的while循环结构，同样得出数值的统计结果。同时，这段程序代码通过数值个数来控制循环，可以统计的数值范围比较大，可以统计包括负数在内的统计信息。



while循环和for循环都是比较常见的循环结构，但是两个循环结构还是有区别的。其中最明显的区别在于，while循环的执行次数是不确定的，而for循环的执行次数是确定的。

7.4.7 综合实例

前面已经介绍过，对于比较复杂的MATLAB程序，经常需要将前面章节介绍的程序结构综合起来使用，才能解决复杂的问题。

例 7.19 在MATLAB中，通过程序来演示小球的抛物线轨迹。

step 1 分析小球的抛物线轨迹模型。假定用户抛小球的速度，也就是小球的初始速度是v_o，以及小球的抛射初始角度是θ。根据基础的物理知识，小球在水平和垂直方向上的速度分量分别为：

$$\begin{cases} v_{xo} = v_o \cos\theta \\ v_{yo} = v_o \sin\theta \end{cases}$$

在本实例中，程序代码需要求解的是抛物线轨迹上水平距离的最长距离。根据相关知识，其距离的求解公式如下：

$$\begin{cases} t = -\dfrac{2v_{yo}}{g} \\ x_{max} = v_{xo}t \end{cases}$$

在上面的公式中，g代表的是重力加速度，在本实例中该参数选择的数值为 -9.82。而对应的，小球在垂直方向上的最高距离为：

$$y_{max} = \frac{v_{yo}^2}{2g}$$

step 2 根据本实例的要求，可以输入抛射小球的初始速度，然后得出相应的计算数据。单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
% Script file ball.m
%
% Purpose:
%    This program calculates the distance traveled by a ball
%  thrown at a specified angle "theta" and a specified velocity
%  "vo" form a point,ignoring air friction .It calculates the angle
%  yeilding maximum range,and also plots selected trajectories.
%
% Define variables:
%   conv       degrees to radians conv factor
%   grav       The gravity accel
%   ii,jj      Loop index
%   index      The maximum range in array
%   maxangle   The angle that gives the maximum range
%   maxrange   Maximum range
%   range      range for a specified angle
%   time       Time
%   theta      Initial angle
%   fly_time   the total trajectory time
%   vo         The initial velocity
%   vxo        x-component of the initial velocity
%   vyo        y-component of the initial velocity
%   x          x-position of ball
%   y          y-position of ball

%  定义常数数值
conv=pi/180;
grav=-9.82;
vo=input('Enter the initial velocity:');

range=zeros(1,91);

% 计算最大的水平距离
for ii=1:91
     theta=ii-1;
     vxo=vo*cos(theta*conv);
     vyo=vo*sin(theta*conv);
     max_time=-2*vyo/grav;
     range(ii)=vxo*max_time;
end

%显示计算水平距离的列表
fprintf('Range versus angle theta:\n');
for ii=1:5:91
    theta=ii-1;
    fprintf('%2d %8.4f\n',theta,range(ii));
end

% 计算最大的角度和水平距离
[ maxrange index] =max(range);
```

```
maxangle=index-1;
fprintf('\n Max range is %8.4f at %2d degrees.\n',maxrange,maxangle);

% 绘制轨迹图形
for ii=5:10:80
     theta=ii;
     vxo=vo*cos(theta*conv);
     vyo=vo*sin(theta*conv);
     max_time=-2*vyo/grav;
     %  计算小球轨迹的x, y坐标数值
     x=zeros(1,21);
     y=zeros(1,21);
     for jj=1:21
         time=(jj-1)*max_time/20;
         x(jj)=vxo*time;
         y(jj)=vyo*time+0.5*grav*time^2;
     end
     plot(x,y,'g');
     if ii==5
         hold on;
     end
end
% 添加图形的标题和坐标轴名称
title('\bf Trajectory of Ball vs Initial Angle\theta');
xlabel('\bf\itx \rm\bf(meters)');
ylabel('\bf\ity \rm\bf(meters)');
axis([ 0 max(range)+5 0 -vo^2/2/grav] );
grid on;

% 绘制最大水平的轨迹图形
vxo=vo*cos(maxangle*conv);
vyo=vo*sin(maxangle*conv);
max_time=-2*vyo/grav;
     %  Calculate the (x,y) position
     x=zeros(1,21);
     y=zeros(1,21);
     for jj=1:21
         time=(jj-1)*max_time/20;
         x(jj)=vxo*time;
         y(jj)=vyo*time+0.5*grav*time^2;
     end
     plot(x,y,'r','Linewidth',2);
     hold off
```

step 3 单击M文件编辑器中的"保存"按钮，将上面的程序代码保存为"ball.m"，得到的结果如图7.24所示。

step 4 返回到MATLAB的命令窗口，输入"ball"，然后按"Enter"键，根据程序代码的提示，依据不同的数值情况，得到的结果如下：

```
>> ball
Enter the initial velocity:20
Range versus angle theta:
 0    0.0000
 5    7.0732
```

```
10   13.9316
15   20.3666
20   26.1828
25   31.2034
30   35.2760
35   38.2767
40   40.1144
45   40.7332
50   40.1144
55   38.2767
60   35.2760
65   31.2034
70   26.1828
75   20.3666
80   13.9316
85    7.0732
90    0.0000
 Max range is  40.7332 at 45 degrees.
```



图 7.24　保存程序代码

除了上面的数值结果之外，MATLAB 还会绘制相应的图形结果，如图 7.25 所示。

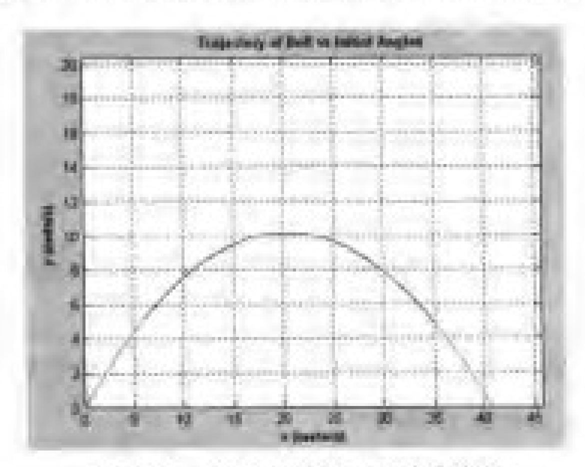


图 7.25　初始速度为 20 时的轨迹

step 5 修改初始速度数值，将其改为 45，得到的结果如下：

```
>> ball
```

```
Enter the initial velocity:45
Range versus angle theta:
 0    0.0000
 5   35.8083
10   70.5286
15  103.1059
20  132.5504
25  157.9674
30  178.5847
35  193.7757
40  203.0790
45  206.2118
50  203.0790
55  193.7757
60  178.5847
65  157.9674
70  132.5504
75  103.1059
80   70.5286
85   35.8083
90    0.0000
  Max range is 206.2118 at 45 degrees
```

同时，MATLAB会给出对应的图形结果，如图7.26所示。

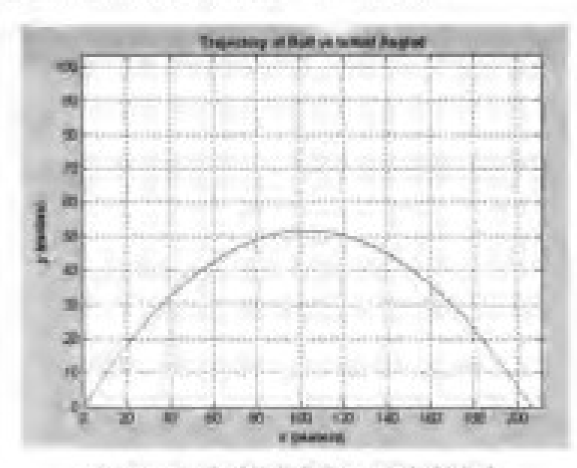


图7.26　初始速度为45时的轨迹



在上面比较综合的实例中，在程序代码中多次用到了if分支结构和for循环结构，功能是实现多种情况下的小球运行的轨迹，读者可以在MATLAB环境中自行运行该程序代码。

7.5 MATLAB的控制语句

在使用MATLAB设计程序时，经常遇到提前终止循环、跳出子程序、显示错误信息等情况，因此还需要其他的控制语句来实现上面这些功能。在MATLAB中，对应的控制语句有continue、break、return、echo等。

7.5.1 continue 命令

在 MATLAB 中，该命令的功能是结束程序的循环语句，也就是跳过循环体中还没有执行的语句。其调用格式比较简单，直接在程序中写出 continue 语句就可以了。下面使用一个简单的实例，说明 continue 命令的使用方法。

例 7.20 使用简单的案例说明 continue 命令的使用方法。

step 1 单击 MATLAB 命令窗口工具栏中的 □ 按钮，打开 M 文件编辑器。在 M 文件编辑器中输入下面的程序代码：

```
for ii=1:9
    if ii==3
        continue
    end
    fprintf('ii=%d\n',ii);
%     if ii==5
%           break
%     end
   end
disp('The end of loop')
```



之所以在上面的程序代码中的第二段代码前面添加了%，是为了首先将其当作注释，不运行这段代码，在后面的程序中将注释符号删除后就可以重新运行了。

step 2 将上面的代码保存为 "break_continue.m" 文件，在 MATLAB 的命令窗口中输入 "break_continue"，然后按 "Enter" 键，就可以得到对应的结果如下：

```
>> break_continue
ii=1
ii=2
ii=4
ii=5
ii=6
ii=7
ii=8
ii=9
The end of loop
```

step 3 打开 "break_contine.m" 文件，在编辑器中修改其代码，得到的结果如下：

```
for ii=1:9
    if ii==3
        continue
    end
    fprintf('ii=%d\n',ii);
   if ii==5
          break
         end
   end
disp('The end of loop')
```

上面的程序代码相当于将前面的注释符号删除，然后保存这种修改即可。

step 4 在MATLAB的命令窗口中输入"break_continue"，然后按"Enter"键，就可以得到对应的结果如下：

```
>> break_continue
ii=1
ii=2
ii=4
ii=5
The end of loop
```



在上面程序代码中使用了break语句，其功能就是跳出相应的程序代码，将在下面的章节中详细介绍。

7.5.2 break 命令

在MATLAB中，break命令的功能在于终止本次循环，跳出最内层的循环，而不必等到循环的结束，而是根据条件退出循环，常常和if语句结合起来运用来终止循环。

例 7.21 在MATLAB中寻求Fibonacci数组中第一个大于700的元素以及其数组标号。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
n=50;a=ones(1,n);
for i=3:n
     a(i)=a(i-1)+a(i-2);
     if a(i)>=700
          a(i)
          break;
     end
end
i
```

step 2 将上面的代码保存为"Fib.m"文件，然后在MATLAB命令窗口中输入"Fib"，按"Enter"键，就可以得到对应的结果如下：

```
>> Fib
ans =
   987
i =
    16
```

从上面的结果中可以看出，在Fibonacci数组中第一个大于700的数值是987，其对应的数组标号是16。

7.5.3 return 命令

在通常情况下，当被调函数执行完后，MATLAB会自动地把控制转至主调函数或者指定窗口。如果在被调函数中插入return命令后，可以强制MATLAB结束执行该函数并把控制转出。

return命令可使正在运行的函数正常退出，返回调用它的函数继续运行，经常用于函数的末尾用来正常结束函数的运行。

在MATLAB的内置函数中，很多函数的程序代码中引入了return命令，下面引用一个简要的det函数代码如下：

```
function d = det(A)
%DET det(A) is the determinant of A.
if isempty(A)
    d = 1;
    return
else
    ...
end
```

在上面的程序代码中，首先通过函数语句来判断参数A的类型，当A是空数组时，直接返回d=1，然后结束程序代码。

7.5.4 input 命令

在MATLAB中，input命令的功能是将MATLAB的控制权暂时交给用户，然后，用户通过键盘输入数值、字符串或者表达式，通过按“Enter”键将输入的内容输入到工作空间中，同时将控制权交还MATLAB，其常用的调用格式如下：

- user_entry = input('prompt')　将用户键入的内容赋给变量 user_entry
- user_entry = input('prompt','s')　将用户键入的内容作为字符串赋给变量 user_entry



对于上面的第一个调用格式，可以输入数值、字符串、元胞数组等各种形式的数据，第二个调用格式，无论输入怎样的变量，都会以字符串的形式赋给变量user_entry。

例 7.22　在 MATLAB 中演示如何使用 input 函数。

step 1　单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function test_input()
reply = input('Do you want more? Y/N [ Y] : ','s');
if isempty(reply)
    reply = 'Y';
end
if reply=='Y'
    disp('you have selected more information');
else
    disp('you have selected the end');
end
```

step 2　将上面的代码保存为“test_input.m”文件，然后在MATLAB命令窗口中输入“test_input”，按“Enter”键，就可以得到对应的结果如下：

```
>> test_input
Do you want more? Y/N [ Y] :
you have selected more information
>> test_input
```

```
Do you want more? Y/N [ Y] : N
you have selected the end
>> test_input
Do you want more? Y/N [ Y] : Y
you have selected more information
```



在上面的程序代码中，使用isempty来接收用户输入的Enter键，当用户什么字符串都不输入的时候，将当作用户输入的是Y。

7.5.5 keyboard命令

在MATLAB中，将keyboard命令放置到M文件中，将停止文件的执行并将控制权交给键盘。通过提示符K来显示一种特殊状态，只有当用户使用return命令结束输入后，控制权才交还程序。在M文件中使用该命令，对程序的调试和在程序运行中修改变量都会十分便利。

例7.23 在MATLAB中，演示如何使用keyboard命令。

在MATLAB的命令窗口中输入下面的内容：

```
>> keyboard
K>>  for ii=1:9
     if ii==3
         continue
     end
     fprintf('ii=%d\n',ii);
   if ii==5
          break
        end
   end
ii=1
ii=2
ii=4
ii=5
K>> return
>>
```

从上面的程序代码中可以看出，当输入keyboard命令后，在提示符的前面会显示k提示符，而当用户输入return后，提示符恢复正常的提示效果。



在MATLAB中，keyboard命令和input命令的不同在于，keyboard命令允许用户输入任意多个MATLAB命令，而input命令则只能输入赋值给变量的数值。

7.5.6 error和warning命令

在MATLAB中，编写M文件的时候经常需要提示一些警告信息。为此，MATLAB提供了下面几个常见的命令。

- error('message') 显示出错信息message，终止程序。
- errordlg('errorstring','dlgname') 显示出错信息的对话框，对话框的标题为dlgname。

✦ warning('message') 显示警告信息 message，程序继续进行。

例 7.24 借用前面小节中的 "stats2.m"，修改其部分程序代码，使用不同的警告样式，查看 MATLAB 的不同错误提示模式。

step 1 在 MATLAB 中，打开 "stats2.m" 文件，在编辑器中修改其程序代码，并将其保存为 "error_message.m" 文件，修改的程序代码如下：

```
sum1=0;sum2=0;
% Get the number of points to input
n=input('Enter the number of points: ');
% Check to see if we have enough input data.
if n<2
    errordlg('Not enough input data','Data Error');
else
..........................................................
% 对于程序的其他部分，在这里全部省略
end
```

step 2 返回 MATLAB 命令窗口，在命令窗口中输入 "error_message"，然后输入数值 1，得到的结果如图 7.27 所示。

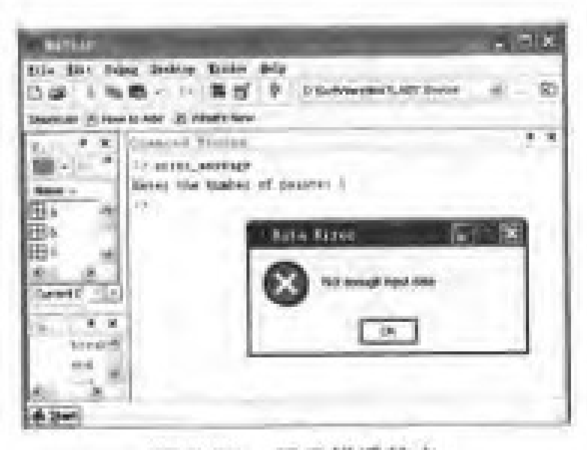


图 7.27 显示错误信息

当用户输入的数值总数小于2时，MATLAB调用错误信息对话框。当用户单击对话框中的"OK"按钮后，将自动退出程序代码。

step 3 打开 "error_message.m" 文件，在编辑器中修改其程序代码，然后保存相应的程序代码，修改的程序代码如下：

```
sum1=0;sum2=0;
% Get the number of points to input
n=input('Enter the number of points: ');
% Check to see if we have enough input data.
if n<2
    error('Not enough input data');
else
     .......................................
      end
```

step 4 返回 MATLAB 命令窗口，在命令窗口中输入 "error_message"，然后输入数值 1，得到的结

果如下：

```
>> error_message
Enter the number of points: 1
??? Error using ==> error_message
Not enough input data
Error in ==> error_message at 24
    error('Not enough input data');
>>
```

step 5 打开 "error_message.m" 文件，在编辑器中修改其程序代码，然后保存相应的程序代码，修改的程序代码如下：

```
sum1=0;sum2=0;
% Get the number of points to input
n=input('Enter the number of points: ');
% Check to see if we have enough input data.
if n<2
    warning('Not enough input data');
else
    ...................................
     end
```

step 6 返回 MATLAB 命令窗口，在命令窗口中输入 "error_message"，然后输入数值 1，得到的结果如下：

```
>> error_message
Enter the number of points: 1
Warning: Not enough input data
> In error_message at 24
```



在上面的程序代码中，演示了 MATLAB 中的不同错误信息方式。其中 error 和 warning 的主要区别在于 warning 命令指示警告信息后继续运行程序。

7.6 程序的向量化概念

在 MATLAB 中，除了前面介绍的常见程序结构之外，还有一种特有的方法——程序的向量化。向量化是一个程序概念，指的是使用向量化的程序代码和语句来替代循环结构。使用向量化可给MATLAB 的程序性能带来质的提高。

在本节中，将介绍如何使用逻辑数组等向量化手段来替代 MATLAB 中的循环结构，提高程序的性能。

7.6.1 程序的向量化

为了向用户介绍程序向量化的概念，使用一个简单的案例来对比循环结构和向量结构化的差别。该案例的目的是，计算某数组元素的平方、平方根和立方根，数组元素是从 1 到 100 的整数。实现该案例的方法可以是循环结构，也可以是向量化程序，下面详细介绍这两种方法。

例 7.25 在 MATLAB 中，使用循环结构和向量化来求解数组的乘幂计算。

step 1 单击 MATLAB 命令窗口工具栏中的 □ 按钮，打开 M 文件编辑器。在 M 文件编辑器中输入下面的程序代码：

```
for ii=1:100
square(ii)=ii^2;
square_root(ii)=ii^(1/2);
cube_root(ii)=ii^(1/3);
end
disp('Square    Square_root   Cube_root')
result=[ square',square_root',cube_root'];
```

step 2 将上面的代码保存为“cycle.m”文件。然后单击 MATLAB 命令窗口工具栏中的 □ 按钮，打开一个新的 M 文件编辑器。在 M 文件编辑器中输入下面的程序代码：

```
ii=1:100;
square=ii.^2;
square_root=ii.^(1/2);
cube_root=ii.^(1/3);
disp('Square    Square_root   Cube_root')
[ square',square_root',cube_root']
```

step 1 将上面的代码保存为“vector.m”文件。然后返回 MATLAB 的命令窗口，在命令窗口中输入"cycle"，然后按“Enter”键，得到的结果如下：

```
>> format short g
>> cycle
Square    Square_root   Cube_root
result =
            1            1            1
            4       1.4142       1.2599
            9       1.7321       1.4422
           16            2       1.5874
           25       2.2361         1.71
           36       2.4495       1.8171
           ......................................
         4225       8.0623       4.0207
         4356        8.124       4.0412
         4489       8.1854       4.0615
         4624       8.2462       4.0817
         4761       8.3066       4.1016
         ............................................
         9409       9.8489       4.5947
         9604       9.8995       4.6104
         9801       9.9499       4.6261
        10000           10       4.6416
```

step 4 在 MATLAB 的命令窗口中输入 "vector"，查看向量化程序的结果：

```
>> vector
Square    Square_root   Cube_root
ans =
            1            1            1
```

```
     4        1.4142        1.2599
     9        1.7321        1.4422
    16             2        1.5874
    25        2.2361          1.71
    36        2.4495        1.8171
........................................................
  4225        8.0623        4.0207
  4356         8.124        4.0412
  4489        8.1854        4.0615
  4624        8.2462        4.0817
  4761        8.3066        4.1016
..............................................................
  9409        9.8489        4.5947
  9604        9.8995        4.6104
  9801        9.9499        4.6261
 10000            10        4.6416
```

从上面的程序段可以看出，两种方法得到的结果是完全相同的，但是两个程序段的效率却有着明显的区别。粗略地讲，使用程序向量化计算速度会比使用for循环的程序结构快15倍。造成这种情况的主要原因在于，对于for循环的程序结构语句，MATLAB每次只能编译其中的一行代码，因此对于上面的程序MATLAB需要编译300行程序代码。而对于后面的向量化代码，MATLAB就只需要编译仅仅4行代码。同时，MATLAB本身就是设置为向量化代码程序的，这样的原因足以提高程序效率。



尽管向量化程序代码在效率和速度上有明显的优势，但是这种效率提高需要占用更多的计算机内存，因为有很多的中间数组产生。但是，从整体上来讲，这种"代价"是值得的，效率上的显著提高使得向量化程序整体上优于循环结构。

7.6.2 向量化和循环结构对比

在上面的步骤中，只是显示向量化程序可以得到和循环结构相同的结果，下面将有必要比较各种不同方法的效率。

例7.26 在MATLAB中，使用三种不同的方法来完成相同的运算。这三种方法依次是使用未初始化的数组和循环结构，使用初始化的数组和循环结构和向量化方法。通过使用MATLAB内置的计时函数来统计三种方法的运行时间。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
% Script file timings.m
%
%  目的：
%  该段程序代码需要使用三种不同方法计算从1到10 000的整数
%  平方的时间：
%  1.不使用初始化数组的数值
%  2.使用初始化的数组数值
%  3.使用向量化

% 定义变量
% ii,jj          循环次数
```

```
% ave1      第一种方法的平均时间
% ave2      第二种方法的平均时间
% ave3      第三种方法的平均时间
% maxcount  循环计算的次数
% square    平方数值的数组

% 不预先设定数组，直接求解结果
maxcount=1;
tic;
for jj=1:maxcount
clear square
for ii=1:10000
    square(ii)=ii^2;
end
end
ave1=(toc)/maxcount;

% 预先设定空数组，计算循环
maxcount=50;
tic;
for jj=1:maxcount
clear square
square=zeros(1,10000);
for ii=1:10000
    square(ii)=ii^2;
end
end
ave2=(toc)/maxcount;

% 使用程序的向量化
maxcount=100;
tic;
for jj=1:maxcount
clear square
ii=1:10000;
    square=ii.^2;
end
ave3=(toc)/maxcount;
%display the results
fprintf('Loop/uninitialized array= %9.5f\n',ave1);
fprintf('Loop/initialized array= %9.5f\n',ave2);
fprintf('vectorized= %9.5f\n',ave3);
```



从上面的程序代码中，通过使用tic和toc命令来统计三种程序结构的运行时间，三种代码的结果是完全相同的，但是运行时间则有明显的差别。

step 2 将上面的代码保存为"timings.m"文件，在MATLAB命令窗口中输入"timings"，然后按"Enter"键，就可以得到对应的结果：

```
>> timings
Loop/uninitialized array=   0.47100
Loop/initialized array=   0.00020
vectorized=   0.00010
```

上面程序代码的运行环境是MATLAB7.0和Pentium 1.60GHz，得到的结果差异还是十分明显的。对于第一种方法，循环次数为1，但是使用的时间却是最多的；第二种方法，循环次数是50，使用的时间却明显减少；第三种方法循环次数最多，占用的时间最少。显然，从程序效率角度来看，最后一种方法（向量化方法）是最佳的。

7.6.3 逻辑数组

前面介绍过MATLAB有两种基础的数据类型——数值和字符串。除了这两种数据类型之外，MATLAB还提供了一种数据类型——逻辑数据。实质上，逻辑数据不能算是一个独立的数据类型，它其实是在标准的数值类型上加了一些"逻辑"的属性。可以通过关系和逻辑表达式创建逻辑数组。在MATLAB中，可以通过使用whos命令来区分逻辑数组和其他的数据数组。

例 7.27 在MATLAB中创建逻辑数组，并通过命令来识别逻辑数组。

step 1 在MATLAB 的命令窗口中输入下面的内容：

```
>> a=[1,2,3;4,5,6;7,8,9];
>> b=a>4;
```

step 2 查看上面步骤得到的数组b，然后识别该数组类型，得到的结果如下：

```
>> b
b =
     0     0     0
     0     1     1
     1     1     1
>> whos
  Name      Size                    Bytes  Class
  a         3x3                        72  double array
  b         3x3                         9  logical array
Grand total is 18 elements using 81 bytes
```



从上面的结果中可以看出，数组b从外观上和数值数组完全相同，但是从类型角度来看，a属于数值数组，但是b属于逻辑数组。

7.6.4 使用logical命令创建逻辑数组

除了使用关系表达式和逻辑表达式之外，可以使用logical命令来创建逻辑数组。灵活使用该命令可以实现很多逻辑功能。

例 7.28 在MATLAB中使用logical命令创建逻辑数组，并进行数组运算。

step 1 在MATLAB 的命令窗口中输入下面的内容：

```
>> a=[1,2,3;4,5,6;7,8,9];
>>b=logical(eye(3));
>>c=a(b);
>> d=a+b;
```

step 2 在命令窗口中依次输入上面的数组名称，得到的结果如下：

```
>> b
b =
     1     0     0
     0     1     0
     0     0     1
>> c
c =
     1
     5
     9
>> d
d =
     2     2     3
     4     6     6
     7     8    10
```

step 3 查看各个数组的数据类型，得到的结果如下：

```
>> whos
  Name      Size                    Bytes  Class
  a         3x3                        72  double array
  b         3x3                         9  logical array
  c         3x1                        24  double array
  d         3x3                        72  double array
Grand total is 30 elements using 177 bytes
```



从上面的程序代码中可以看出，除了c数组之外，其他数组都是数值数组。也就是说，将逻辑数组和其他数值数组运算后，得到的结果将是数值数组。

7.6.5 逻辑数组和向量化

前面之所以花费相当的篇幅来介绍如何使用逻辑数组，主要原因在于逻辑数组可以在程序向量化中起到十分重要的作用，和上面小节类似，用户可以使用逻辑数组和循环语句来完成相同的功能，对比两者的效率。

例 7.29 在 MATLAB 中，使用逻辑数组和循环结构完成相同的数值运算功能：求解大于6000的所有数值的平方根。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
% Script file logical.m
%
% 目的：
%    这段程序代码计算数组中超过6000的平方根
%   使用下面两种不同的方法
%   1.使用循环和if结构
%   2.使用逻辑数组

%定义变量
% ii,jj      循环变量
% ave1       第一种方法的平均时间
```

```
% ave2      第二种方法的平均时间
% maxcount 循环结算的时间次数

%计算循环结构
maxcount=1;
tic
for jj=1:maxcount
    a=1:10000;
    for ii=1:10000
        if a(ii)>6000
            a(ii)=sqrt(a(ii));
        end
    end
end

%使用逻辑数组的方法计算
ave1=(toc)/maxcount;
maxcount=10;
tic;
for jj=1:maxcount
    a=1:10000;
    b=a>6000;
    a(b)=sqrt(a(b));
end
ave2=(toc)/maxcount;

%显示结果
fprintf('Loop if approach= %9.5f\n',ave1);
fprintf('Logical array approach= %9.5f\n',ave2);
```

step 2 将上面的代码保存为"logical.m"文件，在 MATLAB 中的命令窗口中输入"logical"，然后按"Enter"键，就可以得到对应的结果如下：

```
>> logical
Loop if approach=    0.02100
Logical array approach=    0.00100
```

上面程序代码的运行环境是 MATLAB7.0 和 Pentium 1.60GHz，得到的结果差异还是十分明显的。从上面的结果可以看出，逻辑数组的方法比循环结构速度提高了 21 倍。



在本书的这两个小节中，为了对比两种程序方法的性能和程序效率，多次使用了系统的内置函数 toc 和 tic。关于这两种函数的使用方法，可以查看相应的帮助文件。

7.7 脚本和函数

前面已经介绍过，在 MATLAB 中编写的程序类型包括脚本文件和函数文件。这两种类型的文件在很多方面都很相似，但是在语法和使用上还是有很多不同的地方。在 7.1 节就曾经举过简单的案例，编写过脚本文件和函数文件，读者也许已经有了一些直观的感觉，在本节中将详细介绍两种文件。

7.7.1 脚本文件

脚本是一个扩展名为.m 的文件，包括MATLAB的多个命令，与批处理文件类似。在MATLAB的命令窗口中直接输入该文件的名称，MATLAB就会逐一执行文件中的各种命令，效果和在命令窗口中直接输入命令相同。

对于一些比较简单的问题，从命令窗口中直接输入命令进行计算是很简单的事情。但是，当命令的行数增加后，或者控制流的复杂度增加后，直接从命令窗口中输入命令就显得比较烦琐，这个时候脚本文件就比直接输入命令合适。

相对于函数文件，脚本文件的构成比较简单，主要特点如下。

- 脚本文件是一串按用户意图排列而成的MATLAB命令窗口。
- 脚本文件在运行后，所产生的所有变量都驻留在MATLAB的基本工作空间中。只要用户不使用clear命令加以清除，而且只要MATLAB的命令窗口不关闭，这些变量就会保存在基础空间中。

7.7.2 脚本文件实例

例7.30 在MATLAB中，编写绘制三维镂空图形的脚本文件。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码

```
% 脚本文件 loukong.m
t=linspace(0,2*pi,100);
% 旋转母线
r=1-exp(-t/2).*cos(4*t);
% 产生旋转柱面的数据
[x,y,z]=cylinder(r,60);
% 确定第四象限的数据下标
ii=find(x<0&y<0);
% 进行图形剪切
z(ii)=NaN;
surf(x,y,z);
colormap(hsv),shading interp
% 设置光源
light('position',[-3,1,3],'style','local');
% 设置表面反射
material([0.5,0.4,0.3,10,0.3])
```

step 2 将上面的代码保存为"loukong.m"文件，在MATLAB命令窗口中输入"loukong"，然后按"Enter"键，得到的图形如图7.28所示。

step 3 在MATLAB的命令窗口查看工作空间的变量情况，得到的结果如图7.29所示。



从上面的情况中可以看出，在脚本文件中计算得到的中间变量都会保存在MATLAB的工作空间中，不用clear命令清除，则会一直保存在工作空间中。

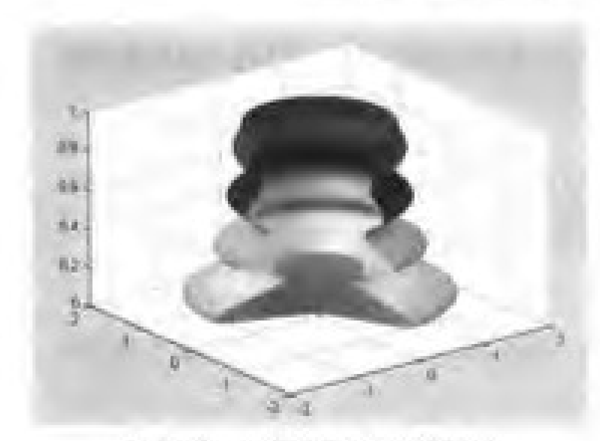


图 7.28　程序代码运行的结果

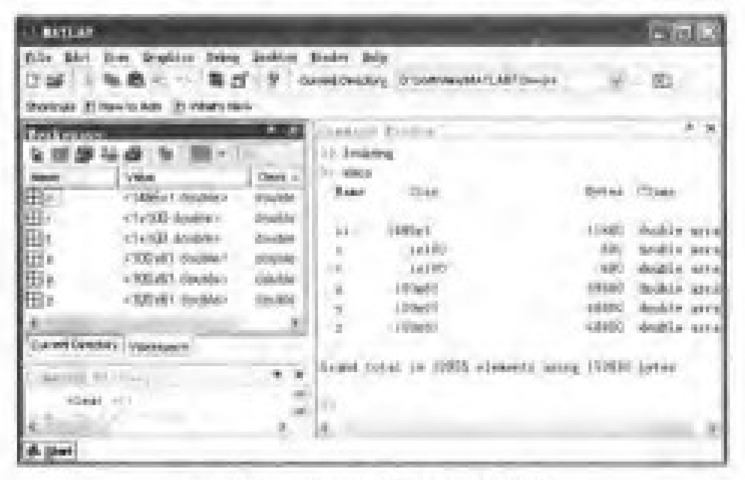


图 7.29　查看工作空间的变量

7.7.3 函数文件

和脚本文件不同，函数文件相当于一个"黑箱"。在外部只能看到传给它的输入参数和输出的结果，内部运作是无法查看的。相对于脚本文件，函数文件的主要特点如下。

- 从形式上看，函数文件的第一行总是以"function"引导的函数申明行。该行还会罗列函数与外界联系的全部"标称"输入和输出变量。如果对输入和输出变量加以限制，那么可以没有输入变量，也可以输入任意数目的变量。
- MATLAB 允许输入比标称数目少的变量，实现对函数的调用。
- 从运行的角度来看，每当函数文件运行时，MATLAB就会专门为其开辟一个临时工作空间。该空间称为函数工作空间，所有的中间变量都会被存放在函数工作空间中。当执行到函数文件的最后一行代码，或者遇到return语句时，就会结束函数的运行，该临时函数空间以及所有的中间变量都会被清除。
- 函数空间会随着具体的M函数文件的调用而产生，随着调用结束而删除。函数空间是相对基础空间独立的、相对的。当MATLAB整个运行期间，可以产生任意多个临时函数空间。
- 如果在函数文件中，调用某个脚本文件，则脚本文件产生的变量都被存放在函数空间之中，而不是存放在基本空间中。

7.7.4 函数文件实例

例 7.31　在 MATLAB 中，编写简单的函数，并使用 ezplot 函数绘制函数的曲线。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function f=myfun(x)
y1=-x.^2+4*x+450;
y2=2.^(sqrt(3*x))+5*x;
f=y1+y2;
```

step 2 将上面的代码保存为"myfun.m"，返回MATLAB的命令窗口中，输入下面的代码：

```
>> fh=@myfun;
>> ezplot(fh,[0,35]);
>> grid
```

step 3 查看图形结果。当输入上面的程序代码后，按"Enter"键，得到图形如图7.30所示。

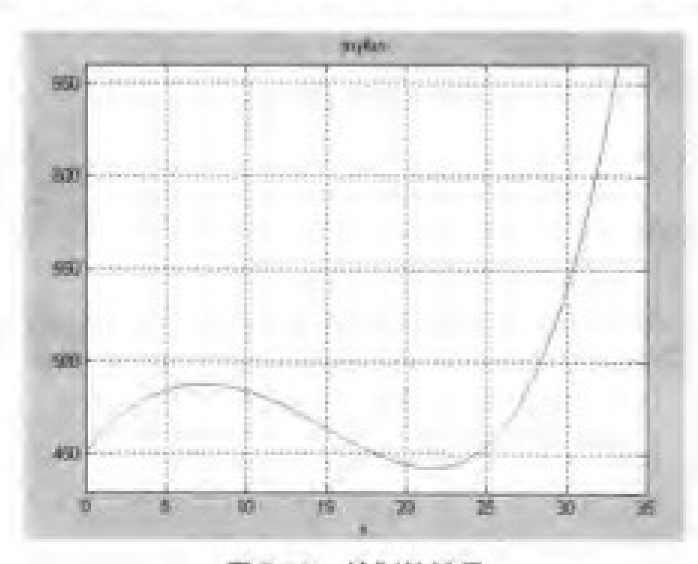


图7.30　绘制的结果

step 4 在MATLAB的命令窗口中查看工作空间的变量情况，得到的结果如下：

```
>> whos
  Name      Size                    Bytes  Class
  fh        1x1                        16  function_handle array
Grand total is 1 element using 16 bytes
```

从上面的查询结果可以看出，函数的中间变量f1、f2和f并没有保存在工作空间中，工作空间中只有一个变量fh。



上面的程序代码中使用了图形句柄和ezplot命令。这些命令都是图形编辑的相关命令，有兴趣的读者可以查看本书的相关章节。

7.7.5 P码文件

在MATLAB中，当一个M文件（脚本文件或者函数文件）被首次调用时，无论在MATLAB命令窗

口中输入文件名，还是使用M文件编辑器打开对应文件，MATLAB都会对该M文件进行语法分析，并把生成的相应内部伪代码存放在内存中。这个内部伪代码简称P码，全称为Psedocode。当再次调用该M文件时，MATLAB会直接运行该文件在内存中的P码文件而不会对原始代码进行重复分析。同时，在MATLAB中，分析器总是把M文件连同被它调用的所有函数M文件一起被变换为P码文件。

P码文件和源代码文件有相同的文件名，扩展名为.p。其运行速度要高于源代码文件，这种速度优势在规模较大的文件中体现得更加明显。

除了MATLAB自动调用之外，可以使用相应的命令来创建P码文件。创建P码文件的主要功能在于保存密码，二进制的P码文件可以执行代码的功能，但是不易于阅读，可以起到很好的保密效果。

创建P码文件的常用命令如下：

```
pcode filename               在当前目录下创建filename.p
pcode. filename -inplace     在filename.m目录下创建filename.p
```

MATLAB还提供对内存中存在的P码文件进行操作的命令，常见命令如下：

```
inmem                    列出内存中的所有P码文件
clear filename           清除内存中的filename.p文件
clear functions          清除内存中的所有P码文件
```



P码文件实例

例 7.32 在MATLAB中，查看内存中的所有P码文件；然后清除所有P码文件，再次查看内存中的P码文件信息。

在MATLAB的命令窗口中输入下面的代码：

```
>> inmem
ans =
    'imformats'
    'workspacefunc'
    'num2str'
    'int2str'
>> clear functions
>> inmem
ans =
    'imformats'
```

如果在MATLAB的运行过程中，曾经打开或者编译某个M文件，则在内存中会保存其对应的P码文件。



变量传递

在编写程序的时候，参数传递一直是十分重要的问题。如何合理安排程序的变量传递直接关系到程序的效率，有时甚至关系到是否能够完成程序功能的问题。在MATLAB中，提供多种函数来实现变量检测、传递，同时也提供“变长度”输入输出变量。灵活使用这些命令可以完成多种复杂的功能，下面详细介绍。

7.8.1 变量检测命令

在MATLAB中，提供多个变量检测命令，用来判断输入和输出变量的个数，相应的调用命令如下：

- n = nargin 在函数体内，用于获取实际的输入变量；
- n = nargin('fun') 获取"fun"指定函数的标称输入变量数目；
- n = nargout 在函数体内，用于获取实际的输出变量；
- n = nargout('fun') 获取"fun"指定函数的标称输出变量数目；
- msgstring = nargchk(minargs, maxargs, numargs) 获取输入变量的数目；
- inputname(n) 在函数体内使用，给出第n个输入变量的实际调用变量名。

例7.33 在MATLAB中，使用nargin函数判断函数的输入变量个数。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function foo(x,y)
if nargin > 2
      errordlg('Input arguments exceeds')
elseif nargin<2
    errordlg('Not enough input arguments')
else
    %foo函数的主体，这里从略
end
```

step 2 将上面的程序代码保存为"foo.m"，返回MATLAB的命令窗口中输入"foo(9)"，得到的结果如图7.31所示。

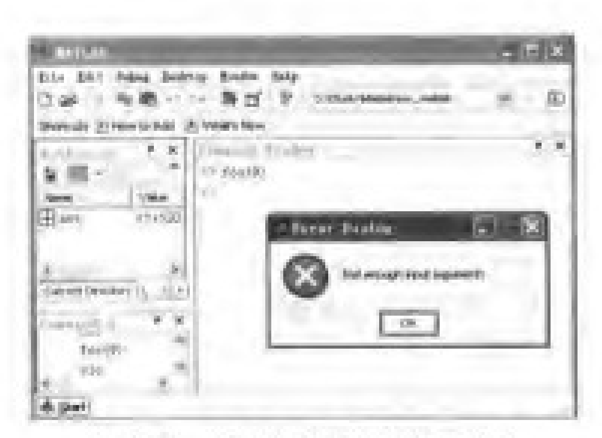


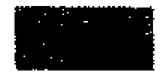
图7.31 输入参数太少的提示信息

step 3 在MALTAB的命令窗口中输入"foo(pi,3,4)"，得到结果如下：

```
>> foo(pi,3,4)
??? Error using ==> foo
Too many input arguments.
```



在MATLAB中，nargin、nargout本身都是内置函数，不是变量，因此不能使用赋值命令对它们进行处理。



“变长度”变量函数

如果比较熟悉MATLAB的plot命令，就会发现一个问题：plot命令的输入变量个数是可变的，可以调用plot(x,y)作为默认格式，也可以使用最完整调用格式：

```
plot(x,y,'PropertyName1',PropertyValue1, 'PropertyName2',PropertyValue2...)
```

这个调用格式允许使用任意多的“属性名/属性值”对，设置绘制图形的属性。因此，plot函数为用户提供可以选择的可变长度变量。除了plot函数之外，MATLAB中还有很多函数可以接收“任意多输入”，返回“任意多输出”。同时，MATLAB也为用户自行编写“变长度”变量函数提供函数：

- varargin “变长度”输入变量列表；
- varargout “变长度”输出变量列表。



“变长度”变量实例

例7.34 在MATLAB中通过“变长度”变量，绘制不同半径、不同圆形的心形图。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function [ xcar,ycar] =drawcardioid(varargin)
%判断变长度变量的个数
error(nargchk(0,3,nargin));
nin = nargin;
%如果用户没有确定参数的数值
if nin==0
% 中心点的坐标数值
    cx1=0; cy1=0;
% 圆形的半径
    r=1;
%如果输入参数的数值个数为1
elseif nin==1
cx1=0; cy1=0;
r=varargin{ 1} ;
%如果输入参数的个数为2
elseif nin==2
cx1=0;
cy1= varargin{ 1} ;
r = varargin{ 2} ;
%如果函数输入参数的个数为3
else
cx1=varargin{ 1} ;
cy1=varargin{ 2} ;
r=varargin{ 3} ;
end

%判断输出参数的个数
if nargout==0
flag=1;
else
flag=0;
end
```

```
% 定义图形的角度变量数值
theta = linspace(0,2*pi,120);
% 计算图形数据点的坐标数值
x0 = r*cos(theta);
y0 = r*sin(theta);
x1 = x0+cx1;
y1 = y0+cy1;
cx2 = 2*x0+cx1;
cy2 = 2*y0+cy1;
x3=x0.*cos(theta)-y0.*sin(theta);
y3=x0.*sin(theta)+y0.*cos(theta);
for k=1:120
x2(k,:) = cx2(k)+x0;
y2(k,:) = cy2(k)+y0;
xx(k)=cx2(k)+x3(k);
yy(k)=cy2(k)+y3(k);
end
xcar = xx;
ycar = yy;
% 绘制图形
if flag
plot(cx1,cy1,'mp','Markersize',6);hold on;
plot(x1,y1);
axis([ cx1-3.2*r cx1+3.2*r cy1-3.2*r cy1+3.2*r]);
axis manual;
hold on
daspect([ 1 1 1]);
set(gcf,'doublebuffer','on');
for k = 1:120
plot(x2(k,:),y2(k,:),'g',xx(k),yy(k),'r*');
grid on;
pause(0.05);
end
end
```

step 2 将上面的代码保存为"drawcardioid.m"，返回MATLAB的命令窗口中，输入"drawcardioid;"，然后按"Enter"键，MATLAB就开始绘制动态图形，如图7.32所示。

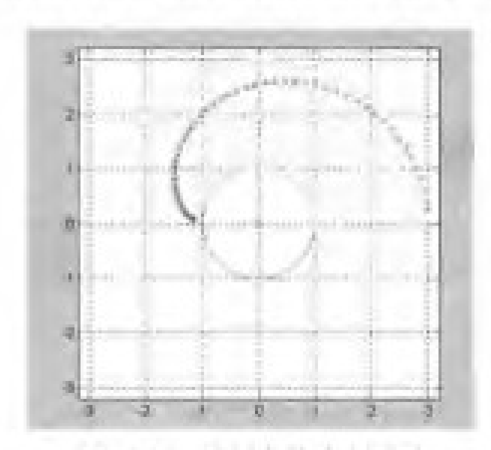


图7.32　绘制中的动态图形

当 MATLAB 完成程序中的循环后，就可以得到绘制完成的图形，如图 7.33 所示。

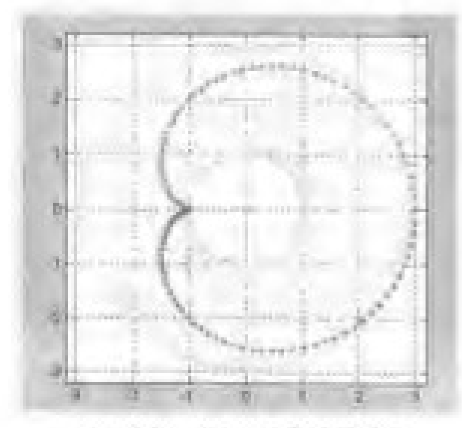


图 7.33 绘制完成的图形



在上面的步骤中，没有输入函数的任何参数，MATLAB 将采用默认的数值，也就是绘制出圆心为原点，半径为 1，外环半径也为 1 的心形图。

step 3 返回 MATLAB 的命令窗口中，输入 "drawcardioid(0,6);"，然后按 "Enter" 键，得到的图形如图 7.34 所示。

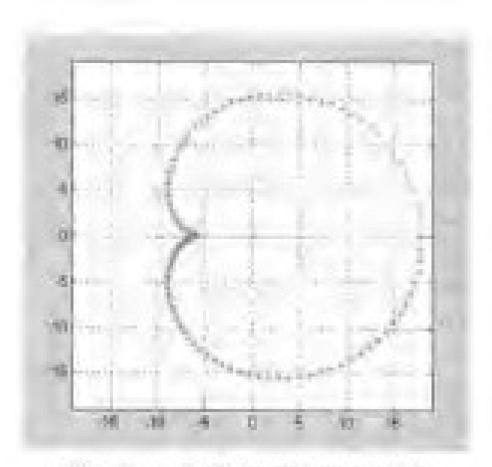


图 7.34 两个参量的绘制图形



根据程序代码，当用户输入函数的两个参数时，圆心的 x 向坐标采用默认的数值 0，第一参数表示圆心的 y 向坐标，第二个参数表示圆的半径。因此，上面的命令绘制的图形是以原点为中心，以 6 为半径的心形图。为了将上面步骤绘制的心形图清除，然后绘制新的心形图，需要在绘制命令之前输入 clf 命令。

step 4 在上面步骤绘制的图形基础上，再绘制一个不同半径的心形图。返回 MATLAB 的命令窗口，输入命令行 "drawcardioid(-1,2,8);"，然后按 "Enter" 键，得到的图形如图 7.35 所示。

根据上面的程序代码，当调用函数时输入 3 个参数，第一个参数和第二个参数分别表示心形图的圆心坐标，在本实例中该坐标为（-1,2）。第三个参数表示心形图的半径，本实例中该数值为 8。因此，上面程序代码绘制的是外层的心形图。

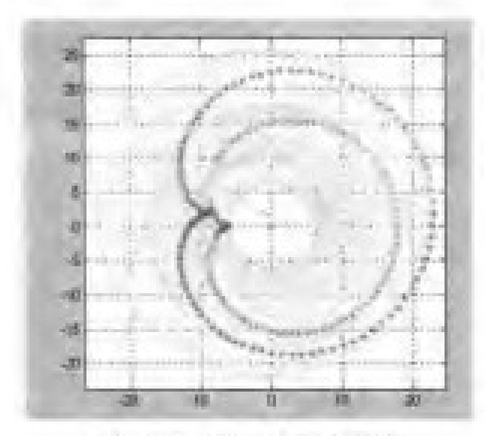


图 7.35　绘制新的心形图

step 5 查看绘制上面图形的数据数组。在 MATLAB 的命令窗口中输入下面的代码：

```
>> [x,y]=drawcardioid(-1,2,8);
>> coordinate=[x',y'];
```

当输入上面的代码后，输入 "coordinate"，按 "Enter" 键，得到的结果如下：

```
>> coordinate
coordinate =
   23.0000    2.0000
   22.9331    3.6876
   22.7331    5.3635
   22.4016    7.0160
   21.9413    8.6337
   21.3562   10.2055
   20.6510   11.7208
   19.8316   13.1694
   18.9048   14.5419
   17.8781   15.8294
   16.7600   17.0240
   15.5594   18.1186
   14.2861   19.1069
   12.9502   19.9838
   11.5623   20.7450
   ..............................     //限于篇幅，省略了部分数据结果
   19.8316   -9.1694
   20.6510   -7.7208
   21.3562   -6.2055
   21.9413   -4.6337
   22.4016   -3.0160
   22.7331   -1.3635
   22.9331    0.3124
   23.0000    2.0000
```

上面的程序代码列出的数据就是函数 drawcardioid 的输出参量。在前面步骤中，没有选择显示输出参量，因此程序并没有得出输出参量。

在编写 M 文件时，函数声明行中的"变长度"变量必须放在一般普通的变量之后。为了帮助读者了解变长度变量定义的工作原理，下面简要介绍 varargin 函数的工作原理。

在 MATLAB 中，varargin 函数本身就是一个元胞数组。当 MATLAB 调用包含 varargin 函数的 M 文件

后，函数输入变量的分配规则是，输入变量按照先后次序逐个对应分配给函数定义的输入变量列表中定义的普通变量，将剩余的输入变量分配到varargin元胞数组中。因此，varargin元胞数组的长度取决于分配到的输入变量数。



当编写M函数文件时，varargin的每个元胞应该当作一个"普通"输入变量处理。varargout的工作原理、规则和varargin相同，差别在于varargout处理的是输出变量和输出变量之间的关系。

7.8.4 跨空间计算表达式的数值

前面已经分别介绍了实现不同工作空间之间变量传递的两种渠道——函数的输入输出变量和全局变量。在MATLAB中，还提供其他方法来实现跨空间的变量传递，在本小节中将介绍跨空间计算表达式数值的命令，详细的调用格式如下：

- evalin(ws,expression) 跨空间计算串表达式的数值；
- evalin(ws,expression1，expression2) 跨空间计算替代串表达式的数值。

例 7.35 在MATLAB中，使用evalin命令实现跨空间传递变量数值。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function y=exeval(theta,arg)
t=(0:theta)/theta*2*pi;
y=subexeval(6,arg);

%子函数代码
function yt=subexeval(theta,arg)
t=(0:theta)/theta*2*pi;argt='theta*exp(i*t)';
switch arg
case {'base','caller'}
   yt=evalin(arg,argt);
case 'self'
   yt=eval(argt);
end
```

step 2 将上面的代码保存为"exeval.m"文件，再次单击MATLAB命令窗口工具栏中的□按钮，打开新的M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
clear;
theta=50;
t=(0:theta)/theta*2*pi;
argt={'base','caller','self'};
for ii=1:3
y1=exeval(10,argt{ii});
subplot(1,3,ii);
plot(real(y1),imag(y1),'m','LineWidth',2)
axis square image;grid on
end
```

将上面的代码保存为"p_exeval.m"文件，该文件是执行前面程序代码的脚本文件。

step 3 返回MATLAB命令窗，输入命令"p_exeval"，按"Enter"键，得到的图形如图7.36所示。

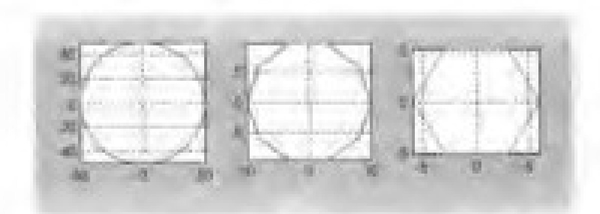


图7.36 程序完成得到的图形结果

当在MATLAB中运行上面的程序代码时，存在三种空间：基础工作空间、主函数（exeval）空间和子函数（subexeval）空间。在每个工作空间中，都存在着不同的参数theta和t的数值。



从上面的结果也可以看出，尽管运算得到的数值都是在子函数中进行的，但是由于参数取自不同空间的数值，因此产生的结果也会互不相同。

在MATLAB中，命令evalin中的ws参数可以选取base和caller两个数值。下面分别介绍两种不同调用格式的机理。

当调用evalin(ws,expression)命令格式时，其执行机理如下：

- 当ws是base时，表示MATLAB运行eval('expression')，将从基本工作空间获得变量数值。
- 当ws是caller时，表示MATLAB在运行eval('expression')，将从主调函数工作空间中获得变量数值。主调函数是相对于被调函数而言的。

当调用evalin(ws,expression1，expression2)命令格式时，其执行机理如下：先从所在函数空间获取数值，用eval('expression1')计算原来字符串表达式；如果计算失败，则再从ws指定的工作空间中获取变量数值，用eval('expression2')计算替代字符串表达式的数值。

7.8.5 跨空间赋值

除了上面介绍的方法外，MATLAB提供另外一种方法来实现不同工作空间之间的变量传递，也就是跨空间赋值命令，其具体调用格式如下：

assignin(ws,'var',val) 跨空间向变量var赋值。

其具体的含义是，将当前工作空间中的变量val赋值给ws指定空间的变量var。

例7.36 在MATLAB中使用assignin命令跨空间赋值。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
%Script file assign.m

%显示信息对话框
prompt = {'Enter image name:','Enter colormap name:'};
```

```
%定义标题
title = 'Image display - assignin example';
lines = 1;
%定义变量的数值
def = {'my_image','hsv'};
%获取答案
answer = inputdlg(prompt,title,lines,def);
assignin('base','imfile',answer{1});
assignin('base','cmap',answer{2});
```

step 2 将上面的代码保存为"assign.m"文件，返回MATLAB的命令窗口，输入"assign"，然后按"Enter"键，得到的结果如图7.37所示。

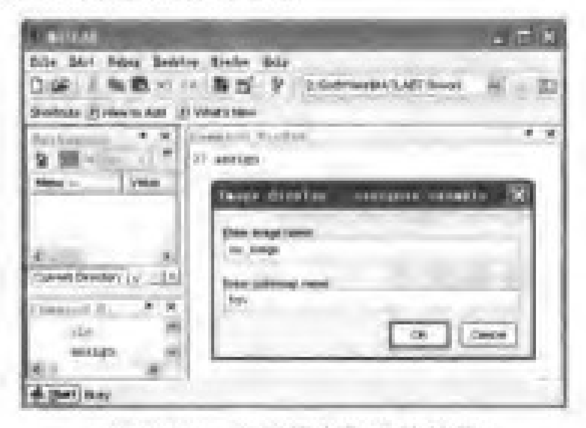


图7.37 显示程序代码的结果

step 3 单击上面对话框中的"OK"按钮，在MATLAB命令窗口中输入whos命令，查看程序得到的变量，如图7.38所示。

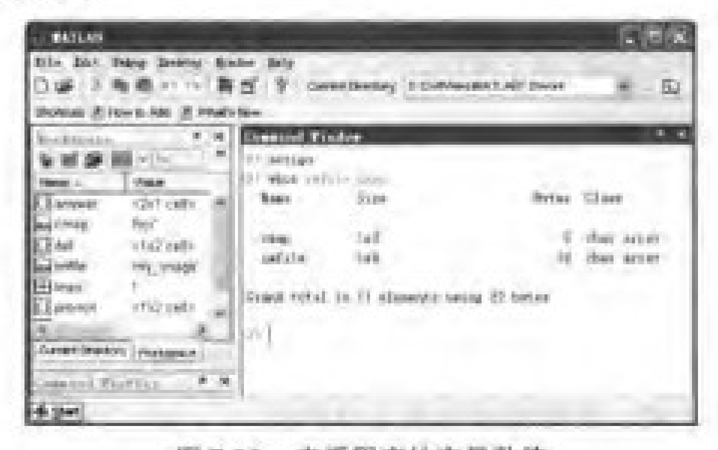


图7.38 查看程序的变量数值

在上面的程序代码中，相关的程序变量都是用户在程序语句中自行定义的，只有变量imfile和cmap是通过assignin命令赋值得到的。

7.9 字符串演算函数

在MATLAB中，各种命令、表达式、语句以及由它们综合组成的M文件，是实现各种计算目的常见方法。为了提高计算的灵活性，MATLAB还提供一种利用字符串进行计算的能力。利用字符串可以构成函数，可以在运行中改变所执行的命令，可以被泛函命令调用，实现比较复杂的运算。



inline 函数

在 MATLAB 中，和字符串运算相关的函数主要有 feval、eval 和 inline 函数等。关于 eval 和 feval 函数的具体使用方法，请查阅“数值计算”章节的相关内容。在本小节中，将详细介绍内联函数 inline 的使用方法。

在 MATLAB 中，内联函数 inline 的调用格式如下：

- g = inline(expr) 将字符串表达式转换为输入变量自动生成的内联函数；
- g = inline(expr,arg1,arg2,...) 将字符串表达式转换为 arg1、arg2 输入变量自动生成的内联函数；
- g = inline(expr,n) 将字符串表达式转换为 x、p1、p2...pn 输入变量自动生成的内联函数。

MATLAB 还提供和 inline 函数相关的处理函数，主要有下面的函数：

- vectorize(inlilne_fun) 使内联函数适合数组运算的法则；
- char(inline_fun) 给出内联函数计算公式。

使用 inline 函数求解零点

例 7.37 在 MATLAB 中，求解超越函数 $f(t)=\sin^2 t\cdot e^{-at}-b|t|$ 的所有零点。

step 1 单击 MATLAB 命令窗口工具栏中的 □ 按钮，打开 M 文件编辑器。在 M 文件编辑器中输入下面的程序代码：

```
% 创建 inline 函数
y=inline('sin(t)^2*exp(-a*t)-b*abs(t)','t','a','b');
% 定义所有的变量
a=0.2;b=0.6;t=-10:0.01:10;
% 向量化 inline 函数
y_char=vectorize(y);
% 计算函数的数值
Y=feval(y_char,t,a,b);
% 绘制函数图形
clf,plot(t,Y,'r','Linewidth',2);hold on,plot(t,zeros(size(t)),'k');
xlabel('t');ylabel('y(t)'),grid,hold off
```

step 2 将上面的代码保存为“inline_zero.m”文件，返回 MATLAB 命令窗口，输入“ainline_zero”，然后按“Enter”键，得到的结果如图 7.39 所示。

step 3 获得函数的零点初始近似数值。上面的步骤显示函数零点的大致分布，现在需要获得近似初始数值。在 MATLAB 命令窗口中输入下面的命令：

```
[r,t]=ginput(5);
```

MATLAB 进入交互界面，可以动态地选取零点的近似数值，如图 7.40 所示。

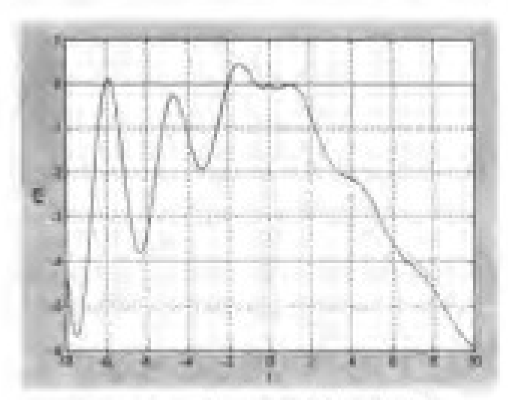


图 7.39　查看函数的零点分布

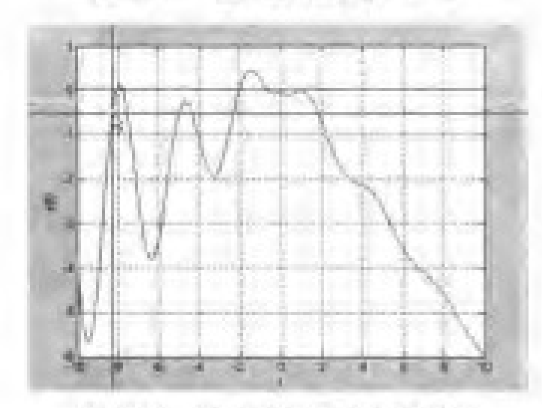


图 7.40　动态选取零点近似数值

step 4 查看选取的结果。当选取图形中的 5 个数据点后，MATLAB 自动结束上面的程序代码。可以在 MATLAB 的命令窗口中查看数据点的坐标，得到的结果如下：

```
>> r
r =
   -8.3180
   -7.5346
   -2.2350
   -0.8065
    0.3917
```

step 5 得到函数的精确零点数值。在 MATLAB 命令窗口中输入下面的程序代码：

```
>> for ii=1:5
[s(ii),m(ii),exitflag]=fzero(y,r(ii),[],a,b);
end
```

step 6 查看计算结果。在上面的程序代码中，s 表示函数准确的零点，m 是对应的函数数值，用户可以在 MATLAB 中查看其相应的数值如下：

```
>> s
s =
   -8.0386   -7.7420   -2.0222   -0.6010   -0.6010
>> m
m =
  1.0e-015 *
   -0.8882         0    0.2220         0         0
```



在上面的程序代码中，最后两个零点的数值都是 -0.6010，而不是0。根据函数的定义，f(0)=0，因此0应该是函数中的一个零点。MATLAB之所以没有查出这个零点，是因为函数图形在 x=0 的地方没有穿越坐标轴。

7.9.3 使用 inline 函数绘制图形

例 7.38 在 MATLAB 中，绘制陀螺运动的图形。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function tlyd;
% tlyd陀螺运动的微分方程
Dfun=inline([ '[ y(2);(1-a)*(y(4))^2*sin(y(1))*cos(y(1))',...
     '-a*y(4)*y(6)*sin(y(1))+b*g*sin(y(1));',...
     'y(4);(a-2)*y(2)*y(4)*cot(y(1))+a*y(2)*y(6)*csc(y(1));',...
     'y(6);y(2)*y(4)*[ sin(y(1))-(a-2)*cot(y(1))*cos(y(1))]',...
     '-a*y(2)*y(6)*cot(y(1))]']','t','y','flag','a','b','g');
% [ t,Y]=ode45(Dfun,[ 0,20],Y0,[ ],a,b,g);
% plot(t,Y)
axis([ -1 1 -1 1 0 1]);
hold on;
text(0,2,0.8,'Start','fontsize',15,'color','g');
grid on;
%获得陀螺侧面的三维数据网格(固连坐标x0,y0,z0为常数)
[ x0,y0,z0]=cylinder(0:.05:.5,60);
axis equal
%以下四句是陀螺底面的数据网格(固连坐标cx0,cy0,cz0为常数)
Q=linspace(0,2*pi,60);
cx0=0.5*cos(Q);
cy0=0.5*sin(Q);
cz0=ones(1,60);
phi=0;thi=pi/6;psi=0;
%陀螺初始位置侧面数据(静坐标x,y,z随陀螺的运动而变化)
[ x,y,z]=zbbh(x0,y0,z0,thi,phi,psi);
%陀螺初始位置底面数据(静坐标cx,cy,cz随陀螺的运动而变化)
[ cx,cy,cz]=zbbh(cx0,cy0,cz0,thi,phi,psi);
%画初始位置陀螺侧面并获取所画图形的图柄
h1=surf(x,y,z);
colormap(winter);
%画初始位置陀螺的底面并获取所画图形的图柄
h2=fill3(cx,cy,cz,[1 0.62 0.40]);
%在继续执行之前,暂停0.5秒
pause(0.5);
%陀螺的参数, R为圆锥底面半径, h为高, p为陀螺材质的密度, g为重力加速度
R=1;h=2;p=1;g=9.8;
a=2/(4*h^2/R^2+1); b=5*p*h/(4*h^2+R^2);
%用函数句柄@tlydfun调用描述陀螺运动微分方程的函数tlydfun
[ t,u]=ode45(Dfun,[ 0:0.1:25],[ thi;0;phi;0;psi;75],[ ],a,b,g);
for i=1: length(u);
     %求陀螺新位置侧面数据
     [ x,y,z]=zbbh(x0,y0,z0,u(i,1),u(i,3),u(i,5));
     %求陀螺新位置底面数据
```

```
        [cx,cy,cz]=zbbh(cx0,cy0,cz0,u(i,1),u(i,3),u(i,5));
        % 画新位置陀螺侧面
        set(h1,'xdata',x,'ydata',y,'zdata',z);
        % 画新位置陀螺底面
        set(h2,'xdata',cx,'ydata',cy,'zdata',cz);
        drawnow;
end;
text(-1.4,0,'End','fontsize',15,'color','g');
%------------ 坐标变换的子函数 -------------------------------%
function [x,y,z]=zbbh(x0,y0,z0,thi,phi,psi)
x=x0*(cos(psi)*cos(phi)-sin(psi)*cos(thi)*sin(phi))...
    +y0*(-sin(psi)*cos(phi)-cos(psi)*cos(thi)*sin(phi))...
    +z0*sin(thi)*sin(phi);
y=x0*(cos(psi)*sin(phi)+sin(psi)*cos(thi)*cos(phi))...
    +y0*(-sin(psi)*sin(phi)+cos(psi)*cos(thi)*cos(phi))...
    -z0*sin(thi)*cos(phi);
z=x0*sin(psi)*sin(thi)+y0*cos(psi)*sin(thi)+z0*cos(thi);
```

完成上面的程序代码后，将其保存为“tlyd.m”文件。在该文件中，tlyd 是主函数，而其中 zbbh 则是被调函数，也就是子函数。

step 2 返回 MATLAB 的命令窗口，输入命令“tlyd”，按“Enter”键，得到的开始图形如图 7.41 所示。

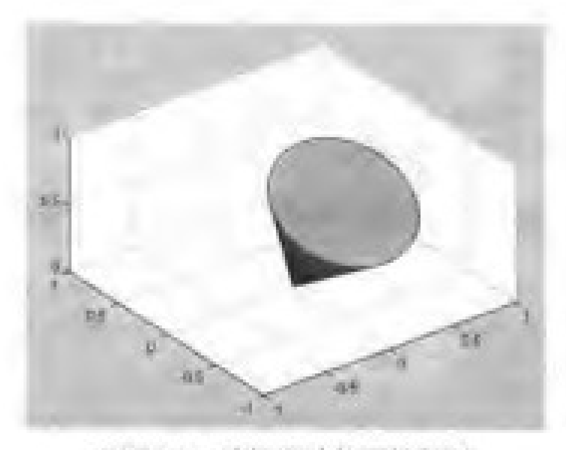


图 7.41　陀螺运动的开始图形

当 MATLAB 开始运行程序后，呈现动态的陀螺运动；当程序循环结束后，就可以得出该程序代码的结束图形，如图 7.42 所示。

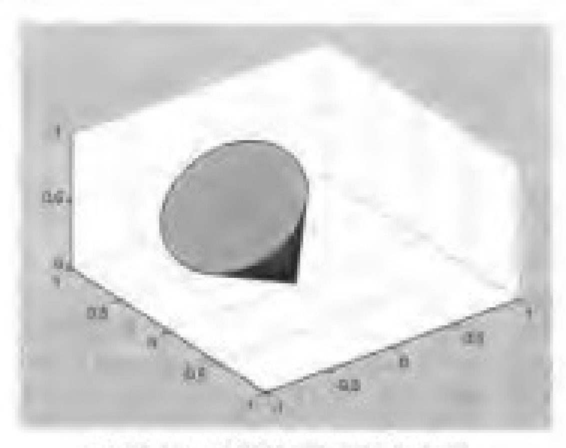


图 7.42　陀螺运动的结束图形

在上面的程序代码中，首先使用 inline 函数定义陀螺运动的微分方程，这就避免了在后面的程序代码中使用字符串带来不必要的麻烦。

 使用 inline 函数求解极值

例 7.39 在 MATLAB 中，使用内联函数求解函数 $f(x,y)=100(y-x^2)^2+(1-x)^2$ 在定义域范围内的极小值。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> ff=inline('100*(x(1)-x(2)^2)^2+(1-x(1))^2','x');
>> x0=[-1.2,1];
```

step 2 查看使用单纯形求解的极小值点。在命令窗口中输入下面的代码：

```
>> [sx,sf,sexit,soutput]=fminsearch(ff,x0)
sx =
    1.0000    1.0000
sf =
  2.0520e-009
sexit =
     1
soutput =
    iterations: 86
     funcCount: 163
     algorithm: 'Nelder-Mead simplex direct search'
       message: [1x196 char]}
```

step 1 查看使用拟牛顿法求解的极小值点。在命令窗口中输入下面的代码：

```
>> [ux,sf,uexit,uoutput,grid,hess]=fminunc(ff,x0)
Warning: Gradient must be provided for trust-region method;
  using line-search method instead.
> In fminunc at 241
Optimization terminated: relative infinity-norm of gradient less than
options.TolFun.
Computing finite-difference Hessian using user-supplied objective
function.
ux =
    1.0000   -1.0000
sf =
  2.3431e-012
uexit =
     1
uoutput =
       iterations: 21
        funcCount: 84
         stepsize: 1
    firstorderopt: 2.1546e-005
        algorithm: 'medium-scale: Quasi-Newton line search'
          message: 'Optimization terminated: relative infinity-norm of
gradient less than options.TolFun.'
grid =
  1.0e-004 *
```

```
   -0.0925
   -0.2155
hess =
  202.0000  400.0241
  400.0241  800.2919
```

step 4 为了方便查看该函数的特性，可以绘制该二元函数的曲面图形。在MATLAB的命令窗口中输入下面的代码：

```
>> x=-1.5:0.1:1.5;y=-1.5:0.1:1.5;
>> [X,Y]=meshgrid(x,y);
>>Z=100*(Y-X.^2).^2+(1-X).^2;
>> surf(X,Y,Z);
>> shading interp
>>colormap hsv;colorbar
```

step 5 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图7.43所示。

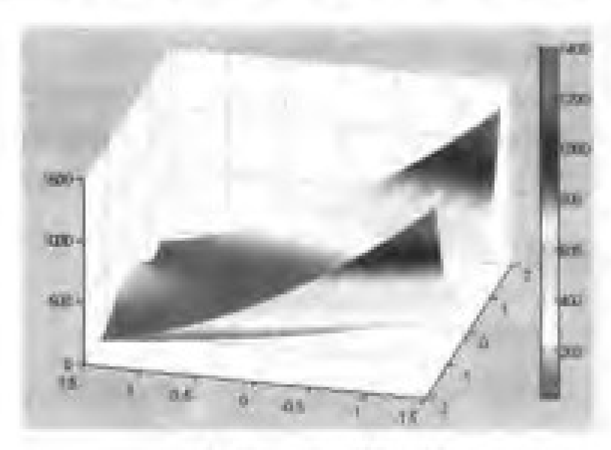


图7.43 函数的图形



从上面的图形中可以看出，该函数有一片浅谷，因此在求解该函数的极小值时，许多算法难以越过这个浅谷。这个函数就是著名的Rosen Brock's "Banana"测试函数。

7.10 程序的调试和剖析

和其他编程语言一样，当使用MATLAB编写M文件的时候，遇到错误（在程序代码中称为Bug）在所难免，尤其是在比较大规模或者多人参与的情况下，掌握程序调试的方法和技巧对提高工作效率很重要。

一般来讲，程序代码的错误主要分为语法错误（Syntax Errors）和逻辑错误（Logical Errors）两种。其中，语法错误通常包括变量名和函数名的误写、标点符号的缺漏或者end的漏写等。对于这类错误，MATLAB会在运行或者P码编译的时候发现，然后终止执行并且报错，用户很容易发现这类错误并改正。

对于逻辑错误，情况则相对比较复杂，处理起来也比较困难。其主要原因如下：逻辑错误一般会涉及算法模型、程序模型是否一致，还涉及编程人员对程序算法的理解等；逻辑错误的表现形态也

较多，如程序运行正常，但是结果异常，或者程序代码不能正常运行而中断。逻辑错误相对于语法错误而言，更难查找错误原因，因为逻辑错误一般是动态的，一旦停止运行，中间变量都会被删除，很难跟踪程序变量的变化情况。

针对上面两种错误类型，本节将详细介绍直接调试法和工具调试法两种常见的调试（Debug）方法。同时，除了介绍如何进行程序调试之外，还介绍如何对程序代码进行解析，以便对程序运行的时间开销进行分析，以改善程序性能。

直接调试法

根据前面章节的介绍，读者已经发现MATLAB语言本身的特点，该语言的向量化程度比较高，程序设计一般都显得比较简单。MATLAB本身的运算能力强，命令系统比较简单，可读性比较强，因此直接调试法往往十分有效。通常采用的直接调试法包括如下手段：

- 经过分析，将重点怀疑语句行或者命令行后面的分号去掉，或者改成逗号，使得运算结果显示在屏幕上面；
- 在有疑问的语句附近，添加显示某些关键变量值的程序语句，查看变量数值；
- 在程序的适当位置添加keyboard命令，将MATLAB执行到相应的程序代码时，会暂停执行程序代码；同时在命令窗口中显示k>>提示符，用户可以查看或者修改变量的数值；在提示符后面输入return命令行后，系统会返回到程序代码中，继续执行原文件。

直接调试法实例

例7.40 在MATLAB中使用直接调试法，调试程序代码。

step 1 单击MATLAB命令窗口工具栏中的 D 按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function f=ballw(K,ki)
% ballw.m 演示红色小球沿一条封闭螺线运动的实时动画
% 仅演示实时动画的调用格式为 ballw(K)
% 既演示实时动画又拍摄照片的调用格式为 f=ballw(K,ki)
% K 红球运动的循环数（不小于 1 ）
% ki 指定拍摄照片的瞬间，取 1 到 1034 间的任意整数
% f 存储拍摄的照片数据，可用 image(f.cdata) 观察照片
t1=(0:1000)/1000*10*pi;x1=cos(t1);y1=sin(t1);z1=-t1;
t2=(0:10)/10;x2=x1(end)*(1-t2);y2=y1(end)*(1-t2);z2=z1(end)*ones(size
(x2));
t3=t2;z3=(1-t3)*z1(end);x3=zeros(size(z3));y3=x3;
t4=t2;x4=t4;y4=zeros(size(x4));z4=y4;
x=[ x1 x2 x3 x4];y=[ y1 y2 y3 y4];z=[ z1 z2 z3 z4];
plot3(x,y,z, 'y','Linewidth',2 ), axis off % 绘制曲线
% 定义 " 线 " 色、 " 点 " 型（点）、点的大小（ 40 ）、擦除方式（ xor）
h=line( 'Color' ,[ 0.67 0 1], 'Marker' , '.' , 'MarkerSize' ,40, 'EraseMode'
, 'xor' );
% 使小球运动
n=length(x);i=1;j=1;
while 1 % 无穷循环
set(h, 'xdata' ,x(i), 'ydata' ,y(i), 'zdata' ,z(i));
drawnow; % 刷新屏幕 <21>
pause(0.0005) % 控制球速 <22>
```

```
i=i+1;
if nargin==2 & nargout==1 % 仅当输入宗量为 2 、输出宗量为 1 时，才拍摄照片
if (i==ki&j==1);f=getframe(gcf); end % 拍摄 i=ki 时的照片 <25>
end
if i>n
i=1;j=j+1;
if j>K; break ; end
end
end
```

step 2 将上面的程序代码保存为 "ballw.m" 文件，在命令窗口中输入 "ballw(2,200)"，得到的图形如图 7.44 所示。

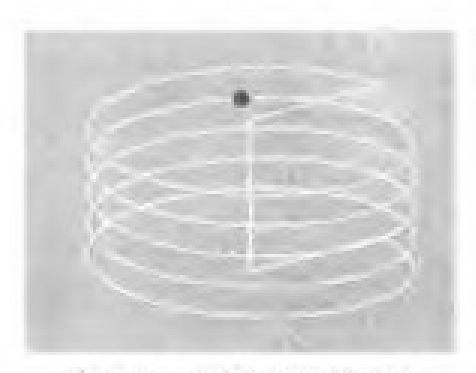


图 7.44　程序运行后的结果

当 MATLAB 完成上面的程序代码后，得到的结果如图 7.45 所示。

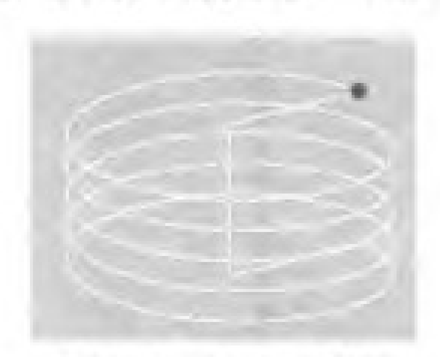


图 7.45　程序结果的图形



说明 从上面的执行情况中可以看出，该程序代码没有任何的语法和逻辑错误。之所以选择该程序代码，是为了在后面的步骤中演示如何直接调试程序代码。

step 3 显示封闭曲线的坐标数值。打开上面步骤保存的 "ballw.m" 文件，将程序代码修改如下：

```
function f=ballw(K,ka)
..................................................// 保持该部分程序代码不变
t3=t2;z3=(1-t3)*z1(end);x3=zeros(size(z3));y3=x3;
t4=t2;x4=t4;y4=zeros(size(x4));z4=y4;
x=[ x1 x2 x3 x4];y=[ y1 y2 y3 y4];z=[ z1 z2 z3 z4];
data=[ x',y',z']                    // 添加显示封闭曲线的坐标数值
...............................................// 保持该部分程序代码不变
if i>n
i=1;j=j+1;
```

```
if j>K; break ; end
end
end
```

修改上面的程序代码后，将程序代码保存为“ballw.m”文件。

step 4 查看程序结果。返回命令窗口，输入命令行“ballw(2,200)”，得到的结果如下：

```
>> ballw(2,200)
data =
    1.0000         0         0
    0.9995    0.0314   -0.0314
    0.9980    0.0628   -0.0628
    0.9956    0.0941   -0.0942
    0.9921    0.1253   -0.1257
    0.9877    0.1564   -0.1571
    0.9823    0.1874   -0.1885
    0.9759    0.2181   -0.2199
    0.9686    0.2487   -0.2513
    0.9603    0.2790   -0.2827
  ..........................................    //限于篇幅，省略了部分数据
         0         0  -15.7080
         0         0  -12.5664
         0         0   -9.4248
         0         0   -6.2832
         0         0   -3.1416
         0         0         0
         0         0         0
    0.1000         0         0
    0.2000         0         0
    0.3000         0         0
    0.4000         0         0
    0.5000         0         0
    0.6000         0         0
    0.7000         0         0
    0.8000         0         0
    0.9000         0         0
    1.0000         0         0
```

从上面的程序结果中可以看出，当在程序代码中添加一个简单的语句“data=[x',y',z']”后，就可以在程序代码执行的过程中，查看封闭曲线的所有坐标值数值。如果程序结果中封闭曲线不正常，则可以从上面的数据中查看数值的问题。

step 5 显示小球位置的坐标数值。打开上面步骤保存的“ballw.m”文件，将程序代码修改如下：

```
function f=ballw(K,ki)
..............................................//保持该部分代码不变
while 1 % 无穷循环
set(h, 'xdata' ,x(i), 'ydata' ,y(i), 'zdata' ,z(i));
bw=[ x(i),y(i),z(i)]                          //计算小球位置的坐标数值
..............................................// 保持该部分代码不变
if i>n
i=1;j=j+1;
if j>K; break ; end
end
end
```

修改上面的程序代码后，将程序代码保存为“ballw.m”文件。

step 6 查看程序结果。返回到命令窗口，输入命令行“ballw(2,200)”，得到的结果如下：

```
bw =
    0.2000   -0.0000  -31.4159
bw =
    0.1000   -0.0000  -31.4159
bw =
         0         0  -31.4159
bw =
         0         0  -31.4159
bw =
         0         0  -28.2743
............................................//限于篇幅，省略了部分数据
bw =
    0.6000         0         0
bw =
    0.7000         0         0
bw =
    0.8000         0         0
bw =
    0.9000         0         0
bw =
     1     0     0
```

在上面的步骤中，分别使用简单程序代码查看关键的程序数据，如果在程序运算过程中出现问题，则可以从上面的程序数值中查找相应的问题。

工具调试法

上面演示了如何使用直接调试法来调试程序代码，但是如果函数文件规模较大，文件的内嵌复杂，或者有较多函数、子函数、私有函数待调用，直接调用法则可能会失败，这个时候则需要使用MATLAB的专门调试工具——调试器。

MATLAB自带的M文件编辑器同时也是程序代码的编辑器，可以在M文件编辑器中输入程序代码后，直接在其中进行调试，显得方便和直观。

关于M文件编辑器的功能在前面介绍过，在本小节中将需要介绍该编辑器的调试功能键和菜单选项。为了显示M文件编辑器的调试功能，有必要调用某个M文件。在本小节中，调用的是7.1.2小节中编写的“test_sort.m”文件，调用的结果如图7.46所示。

在上面的文件调试器中，当设置M文件进入调试阶段时，M文件编辑器提供的调试功能键都会被激活，表示可以使用这些功能键来调试程序代码。为了介绍方便，M文件的调试功能键图形如图7.47所示。

将上面的调试功能键从左到右依次命令为1、2…7，其对应的功能介绍如下：

- **1号功能键**：功能是断点设置（或者清除），对应的菜单选项为“Debug” ⇨ “Set/clear breakpoints”，快捷键为F12，对应的MATLAB命令为dbstop/dbclear。
- **2号功能键**：功能是清除全部断点，对应的菜单选项为“Debug” ⇨ “Clear Breakpoints in All Files”，对应的MATLAB命令为dbclear all。



图 7.46　M 文件调试器

图 7.47　调试功能键

- **3 号功能键：**功能是单步执行当前命令行，对应的菜单选项为"Debug" ⇨ "Step"，快捷键为 F10，对应的 MATLAB 命令为 dbstep。
- **4 号功能键：**功能是深入被调函数，对应的菜单选项为"Debug" ⇨ "Step In"，快捷键为 F11，对应的 MATLAB 命令为 dbstep in。
- **5 号功能键：**功能是跳出被调函数，对应的菜单选项为"Debug" ⇨ "Step Out"，快捷键为"Shift+F11"，对应的 MATLAB 命令为 dbstep out。
- **6 号功能键：**功能是连续执行命令行，对应的菜单选项为"Debug" ⇨ "Continue"，快捷键为 F5，对应的 MATLAB 命令为 dbcont。
- **7 号功能键：**功能是退出调试模式，对应的菜单选项为"Debug" ⇨ "Exit Debug Mode"，对应的 MATLAB 命令为 dbquit。

从上面的介绍中可以看出，常见的调试功能键都可以从对应的菜单选项中实现。除了上面的功能键之外，MATLAB 还提供其他的调试功能。如果希望修改或者设置条件断点，可以选择菜单选项"Debug" ⇨ "Set/Modify Conditional Breakpoint"，MATLAB 会弹出"MATLAB Editor"对话框，如图 7.48 所示。



图 7.48　设置条件断点

如果希望设置程序代码的暂停设置，可以选择菜单选项"Debug" ⇨ "Stop if Error/Warnings……"，打开对应的对话框，如图 7.49 所示。

在 MATLAB 的 M 文件调试器中，除了上面常见的功能键和菜单选项之外，还有一个"空间堆栈"下拉菜单值得引起注意。在调试文件代码时，可以通过改变它的内容来选取和操作不同工作空间的变量，如图 7.50 所示。


图 7.49 设置程序代码的警告信息

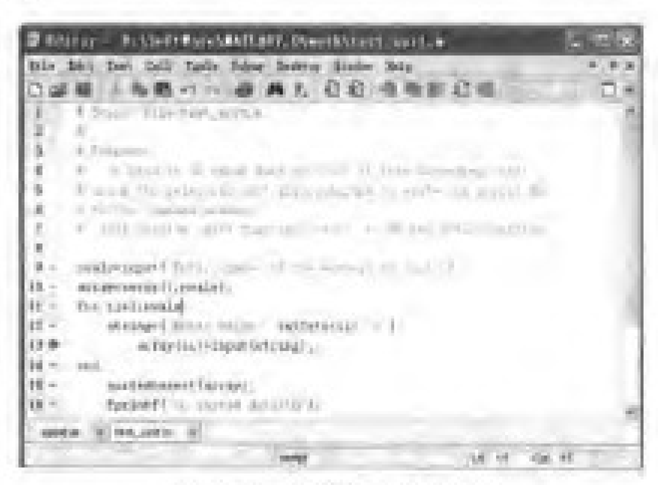

图 7.50 工作空间切换

7.10.4 工具调试法实例

下面通过一个实例来说明如何使用 MATLAB 的工具调试代码程序。

例 7.41 选用 7.1.2 小节中编写的函数文件 "ssort.m" 和脚本文件 "test_sort.m"，演示如何在 MATLAB 调试代码程序。

step 1 打开保存的脚本文件 "test_sort.m"，然后在程序代码的第 13 行添加程序断点，得到的结果如图 7.51 所示。


图 7.51 添加程序的断点

step 2 选择 M 文件编辑器的工具栏中的 "Debug" ⇨ "Run" 命令，或者直接选择快捷键 F5，开始该程序代码的调试工作，得到的结果如下：

```
Enter number of the numbers to sort:2
14       array(ii)=input(string);
K>> return
Enter value 1:2
14       array(ii)=input(string);
K>> return
Enter value 2:3
 sorted data:
  2.0000
```

```
  3.0000
```

从上面的程序运行结果可以看出，程序代码可以不受影响地正常运行，但是在添加程序断点的地方会显示程序语句，并会出现 "K>>" 的提示符，当输入 "return" 命令行后，程序可以继续进行。

step 3 除了可以使用语句 "return" 命令来返回程序代码的调试之外，还可以使用 "深入被调文件" 按钮，查看程序代码的运行情况如下：

```
Enter number of the numbers to sort:3
13          array(ii)=input(string);
Enter value 1:2
14  end
K>> return
13          array(ii)=input(string);
Enter value 2:1
14  end
K>> return
13          array(ii)=input(string);
Enter value 3:5
14  end
K>> return
 sorted data:
  1.0000
  2.0000
  5.0000
```



上面的程序代码只是简单地演示如何使用工具调试法来调试程序，对于其他的调试方法，也许更适合于使用在比较复杂的程序代码中，读者可以自行尝试。

7.10.5 程序剖析

程序调试的主要目的在于对运行不正常的M文件的错误进行定位和纠正，而程序剖析的主要功能在于对M文件的各项命令耗时进行分析，用以提高整个程序代码的性能。特别对于编程的初学者，对程序代码进行剖析，找到程序代码的瓶颈所在，然后有针对性地重新编写程序代码，直到MATLAB的剖析报告表明程序大部分的运行时间被花费在内置函数中，表明该程序代码的性能比较优良。

和程序调试类似，在MATLAB中进行程序剖析也有两种调用方法——命令调用方法和图形界面调用方法。其中，在 MATLAB 中常见的调用命令如下：

- profile on -detail level 开启程序剖析器，并清除以往的剖析记录；
- profile off 暂停程序剖析器的运行；
- profile resume 保留以往的剖析数据，重新启动程序剖析器；
- profile clear 清除以往的剖析记录；
- profile viewer 开启界面式的程序剖析器。



在上面的程序命令中，参数 level 可以选择 mmex、builtin 和 operator 三个数值。它们分别指定剖析数据的统计层次——M文件层，内建函数层和运算命令层。



程序剖析实例

例 7.42 选用自行编写的 M 文件 sierpinskihm.m，对其进行程序代码的剖析。

step 1 打开已经编写完成的“sierpinskihm.m”，其代码显示如下：

```
function sierpinskihm(n);
% Sierpinskinhm海绵
% 调用格式:sierpinskihm(n);
% n为迭代次数
x=0; % x为初始正方形的第一顶点的横坐标
y=0; % y为初始正方形的第一顶点的纵坐标
z=0; % z为初始正方形的第一顶点的竖坐标
d=1; % d为初始正方形的边长
x4=x;
y4=y;
z4=z;
d4=d;
x2=[ ];
y2=[ ];
z2=[ ];
a=[ 0,1,2];
[ x1,y1] =meshgrid(a);
[ x1,z1] =meshgrid(x1,a);
[ y1,z1] =meshgrid(y1,a);
x1=[ x1(1,1:4),x1(1,6:9),x1(2,[ 1,3,7,9] ),x1(3,1:4),x1(1,6:9)];
y1=[ y1(1,1:4),y1(1,6:9),y1(2,[ 1,3,7,9] ),y1(3,1:4),y1(1,6:9)];
z1=[ z1(1,1:4),z1(1,6:9),z1(2,[ 1,3,7,9] ),z1(3,1:4),z1(3,6:9)];
for q=1:n;
    for p=1:length(x);
        x3=x(p)+d/3*x1;
        y3=y(p)+d/3*y1;
        z3=z(p)+d/3*z1;
        x2=[ x2,x3];
        y2=[ y2,y3];
        z2=[ z2,z3];
    end
    d=d/3;
    x=x2;
    y=y2;
    z=z2;
end
axis([ x4,x4+d4,y4,y4+d4,z4,z4+d4] )
P1=[ d,d,d,d,d];
P2=[ 0,d,d,0,0];
P3=[ 0,0,d,d,0];
P4=[ 0,0,0,0,0];
for p=1:length(x);
    patch(x(p)+P1,y(p)+P2,z(p)+P3,z(p)+P3);
    patch(x(p)+P2,y(p)+P4,z(p)+P3,y(p)+P4);
    patch(x(p)+P2,y(p)+P1,z(p)+P3,x(p)+P2);
    patch(x(p)+P2,y(p)+P3,z(p)+P4,y(p)+P3);
    patch(x(p)+P2,y(p)+P3,z(p)+P1,x(p)+P4);
    patch(x(p)+P4,y(p)+P2,z(p)+P3,z(p)+P3);
end
```

```
axis off
set(gcf,'color',ones(1,3))
```

step 2 启动界面式的代码剖析器。选择M文件编辑器中的菜单命令 "Desktop" ⇨ "Profiler"，或者在命令窗口中键入命令行 "profile viewer"，打开性能剖析器，如图7.52所示。



图7.52 程序代码剖析器



在默认情况下，程序代码剖析器在MATLAB中会独立显示。如果希望操作方便，可以选择该界面中的菜单选项 "Desktop" ⇨ "Dock Profiler"，使剖析器嵌在MATLAB的操作桌面上。

step 3 运行代码剖析。在性能剖析器界面中的 "命令输入行（Run this code）" 中，输入运行分析的代码。在本例中输入 "sierpinskihm(3)"，单击 "Start profiling" 按钮，开始代码剖析，如图7.53所示。

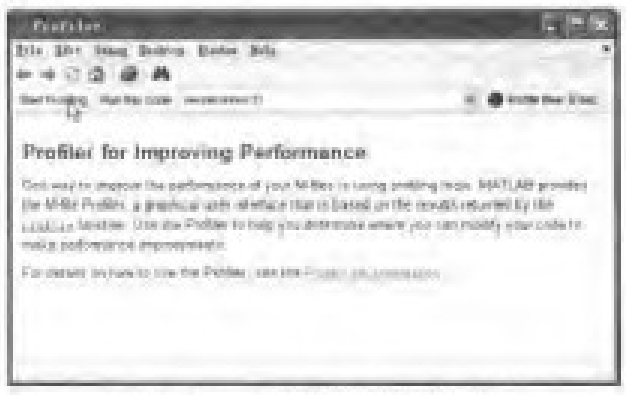


图7.53 运行程序代码的剖析

当单击 "Start profiling" 按钮后，程序的剖析就开始了，这个时候 "Start profiling" 按钮变成灰色，程序将该按钮的功能变成不可用（disabled）；而界面右上侧的运行指示灯则变成 "绿色"，表示程序剖析处于 "工作" 状态，同时，计时器数值连续累加，显示剖析器所经历的工作时间，如图7.54所示。

当程序运行结束时，剖析分析自动暂停，此时 "Start profiling" 按钮恢复原来的颜色，重新恢复所有的功能。运行执行灯也会变成棕色，表示剖析分析处于暂停状态，计时数也可以近似反映被剖析命令的总时间，如图7.55所示。



图 7.54 剖析程序当中



图 7.55 剖析结果

在本实例中，剖析程序代码的时间为15s，同时在上面的结果中显示了关于该程序代码的"分析汇总表（Profile Summary）"。可以在这个比较综合的结果中查看关于该程序的代码剖析结果，当程序运行结束后，会得到该程序的结果，如图 7.56 所示。

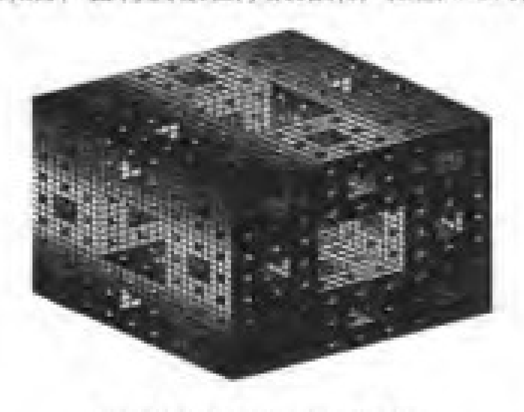


图 7.56 程序运行的结果

step 4 分析代码剖析结果。上面的剖析结果属于比较综合的内容，如果希望了解比较详细或者细节的内容，可以单击界面中带有超链接的文字。在本例中，单击"sierpinskihm"字样，查看对应的详细内容，其中关于消耗时间的内容如图 7.57 所示。

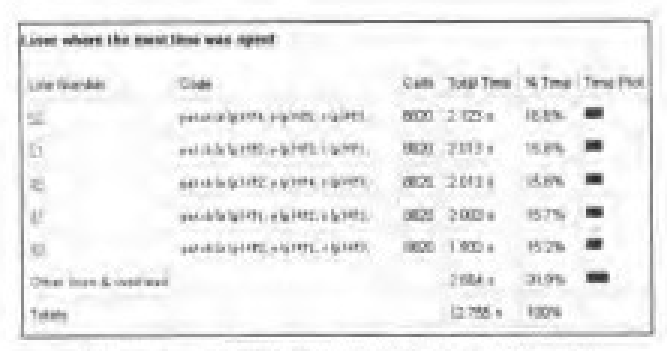


图 7.57 程序代码中最消耗时间的程序行

同时，MATLAB 还会显示关于该程序代码的运行性能列表，如图 7.58 所示。



图 7.58 程序代码的性能分析列表

上面的剖析报告很详细地表明了该段程序代码的"瓶颈"位置，也就是用户可以自行修改的关键程序行。

7.11 小结

本章主要介绍了在MATLAB中进行程序设计的基础内容，主要包括MATLAB变量、表达式、MATLAB的程序结构、控制语句、向量化、变量传递和程序的调试和剖析等，这些内容都是在MATLAB中进行程序设计所需要掌握的基础内容。在后面的章节中，将介绍Simulink仿真系统。

◆ ◆ ◆

第8章　Simulink 仿真系统

本章包括

- Simulink的基础知识
- Simulink的数据类型
- Simulink的基础操作
- Simulink的信号
- Simulink的属性设置
- 线性系统建模
- 非线性系统建模

在MATLAB中，Simulink是用来建模、仿真和分析动态多维系统的交互工具。可以使用Simulink提供的标准模型库或者自行创建模型库，描述、模拟、评价和精化系统行为。同时，Simulink和MATLAB之间的联系十分便捷，可以使用一个灵活的操作系统和应用广泛的分析和设计工具。最后，除了可以使用Simulink建模和仿真之外，还可以通过其他软件包产品来完成更多的分析任务。

在本章中，首先使用一个例子来说明Simulink的仿真创建过程，然后介绍Simulink的工作环境和常见工具。由于Simulink的内容比较繁多，本章将主要介绍关于Simulink的基础知识，包括线性系统、非线性系统和离散系统的建模方法。最后，本章还将介绍关于Simulink分析工具的相关知识，这样，就可以更方便地使用该工具分析Simulink创建的模型对象。关于Simulink的更深层次的应用知识，将在后面的章节中详细介绍。



Simulink 的基础知识

Simulink是一个复杂的应用系统。为了让读者更直观地了解Simulink的使用方法和操作界面方法，在本节将首先介绍关于Simulink的一些基础知识，包括Simulink概述和安装知识，创建方法等。



Simulink 概述

和MATLAB的其他组件（有时也会被称为软件包）相比，Simulink的一个突出特点就是它完全支持图形用户界面（GUI），这样就极大地方便了用户的操作方法。用户只需要进行简单的拖曳操作就可以构造出复杂的仿真模型，它的外观以方块图形的形式来呈现，而且采用分层结构。从建模的角度来看，这种方法可以让用户将主要的精力放在具有创造性的算法和模块结构的设计上，而不用将精力放在算法的实现上。从分析研究的角度来看，Simulink模型可以让用户知道具体环节的动态细节，而且还可以让用户清晰地了解到各系统组件、各子系统、各系统之间的信息交换。

在Simulink环境中，用户可以观察到现实世界中摩擦、风阻等非线性或者随机因素对系统行为的影响，同时可以在仿真过程中改变需要观察的参数数值，观察系统行为的变化。这样，可以摆脱复杂的数学推演和烦琐的程序代码，直接探索各种因素的影响。

在MATLAB7.0中，可以直接在Simulink环境中运行的工具包很多，包括通信、控制、信号、电力等各个领域，所涉及的内容也比较广泛和专业。如果用户合理地使用这些工具包中的内容，可以创建各种复杂的仿真模型，实现各种复杂的功能。在本章后面的部分内容中，将会涉及这些工具包的内容，在对应的地方将详细介绍其内容。

鉴于Simulink的实质功能，本节的内容必定会涉及物理、信号、控制或者工程等知识，为了让读者了解到Simulink的使用方法，在本章的对应部分内容将对这些知识做必要的解释，希望读者在了解背景知识的前提下体验到Simulink仿真的真实和精妙。



在MATLAB6.x版本中，Simulink的版本是5.0。在MATLAB7.0版本中，Simulink的版本是6.0，相对于以前的版本，在功能上有了很大的改进，在本章对应的地方将会作相应的介绍。

8.1.2 安装 Simulink

在本书的第1章中，介绍过如何安装MATLAB，而在MATLAB中，Simulink是一个重要的组件，因此已经安装了关于Simulink的基础组件。为了更加全面地了解Simulink的应用方法，还需要安装Simulink的相关组件。插入MATLAB7.0的光盘，启动安装程序，然后选择自定义安装方法，进入文件选择对话框，选择需要安装的组件，如图8.1所示。

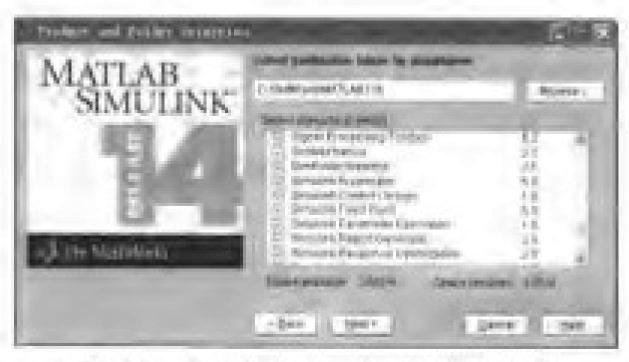


图8.1　选择安装 Simulink 的相关组件

为了更好地了解本章的内容，需要选择安装如下主要的Simulink组件：

- Signal Processing Toolbox
- SimMechanics
- SimPowerSystems
- Simulink Accelerator
- Simulink Contorl Design
- Stateflow
- Real-Time Workshop
- Virtual Reality Toolbox

不用担心上面的组件会重复安装。如果已经在上一次的安装过程中安装了某个组件，而在本次安装过程中再次选择安装该组件，MATLAB会显示如图8.2所示的提示。

在该对话框中，系统提示用户在上一次的安装过程中已经安装了某个（某些）MATLAB组件，而本次用户再次选择安装该组件，系统将提示用户是否将这些组件进行覆盖。如果希望重新安装这些组件，可以单击"Yes to All"按钮。或者单击"Cancel"按钮后，返回选择安装组件的对话框，取消选中这些软件组件，再选择安装。



图 8.2　显示覆盖某组件

8.1.3 启动 Simulink

作为 MATLAB 的重要组件，用户可以使用多种方法来启动 Simulink。其中，最简单的方法就是直接单击 MATLAB 命令窗口中的 Simulink 按钮，如图 8.3 所示。

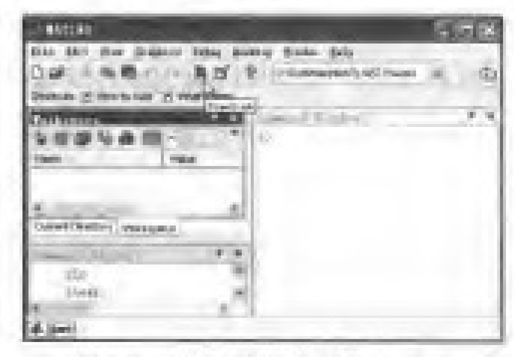


图 8.3　使用按钮启动 Simulink

除了上面的方法外，还可以使用下面两种方法

- 在 MATLAB 的命令窗口中输入命令 ">> Simulink"；
- 选择 MATLAB 编辑栏中的 "File" ⇨ "New" ⇨ "Model" 命令。

使用上面任何一种方式，就可以打开 "Simulink Library Browser" 对话框，在该对话框中可以选择查看各种 Simulink 模块，如图 8.4 所示。



图 8.4　Simulink 模块库浏览器

在上面的浏览器对话框中，可以浏览 Simulink 的常用模块，也可以创建新的模型，打开已经创

建的模块等。因此，该对话框可以认为是Simulink操作的基础版块，在后面的章节中将详细介绍该对话框的使用方法。



如果使用"File" ⇨ "New" ⇨ "Model"命令来打开Simulink系统，MATLAB除了显示"Simulink Library Browser"对话框之外，还会显示新建模块的窗口，在后面的章节中将详细介绍该窗口。

8.1.4 添加Simulink模块

为了让用户直观地了解Simulink的使用方法，在本小节中将使用一个比较简单的实例来说明Simulink的创建过程和步骤。

例8.1 使用Simulink来计算"chirp"信号和正弦函数信号叠加后的信号波形。

step 1 单击"Simulink Library Browser"对话框中的按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "Model"命令，打开一个空白模型窗口，如图8.5所示。



图8.5 新建模型窗口

step 2 选择"Chirp Singal"信号源。选择"Simulink Library Browser"对话框左侧的"Sources"模块库，然后在对话框的右侧选择"Chirp Singal"模块，如图8.6所示。



图8.6 选择"Chirp Singal"信号源



当在“Simulink Library Browser（模块库浏览器）”中选择相应的信号源后，在对话框的上半部分就会显示关于该信号的基础介绍，通过这些基础的介绍大致了解该信号的内容。

step 3 添加“Chirp Singal”信号源。选中“Chirp Singal”模块，按下鼠标左键，将其拖到新建模型窗口中，如图 8.7 所示。



图 8.7 添加“Chirp Singal”信号

step 4 查看添加的“Chirp Singal”信号源。当选择添加的合适位置后，松开鼠标左键，在对应的位置就会显示用户添加的信号模块，如图 8.8 所示。



图 8.8 添加信号模块



当向模型窗口中添加模块后，对话框的名称由“untitled”变为“untitled*”，同时，所添加的模块也会采用默认的设置和大小，用户可以根据需要修改这些属性。

8.1.5 设置模块的属性

延续上面小节的步骤。

step 1 编辑“Chirp Singal”模块的外观属性。选中“Chirp Singal”模块，当模块出现对应的模块柄后，按下鼠标并进行拖动，改变模块大小；选择菜单栏中的“Format” ⇨ “Background Color” ⇨ “Green”命令，将模块的背景颜色设置为绿色，如图 8.9 所示。

step 2 设置“Chirp Singal”模块的参数。双击模块窗口的“Chirp Singal”模块，打开“Block Parameters:Chirp Signal”对话框，设置该模块对应的参数，如图 8.10 所示。

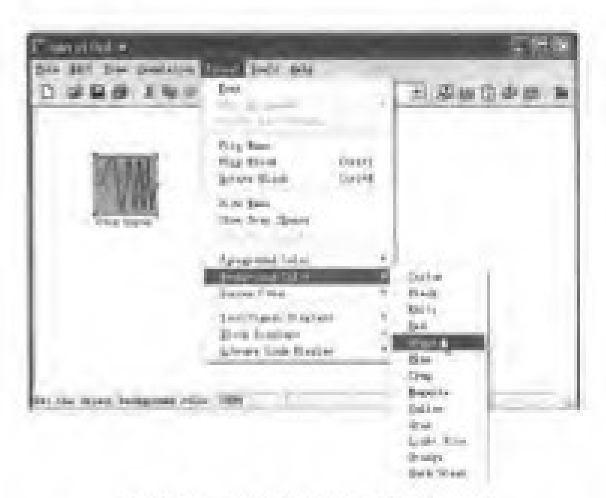


图 8.9　设置模块的外观属性

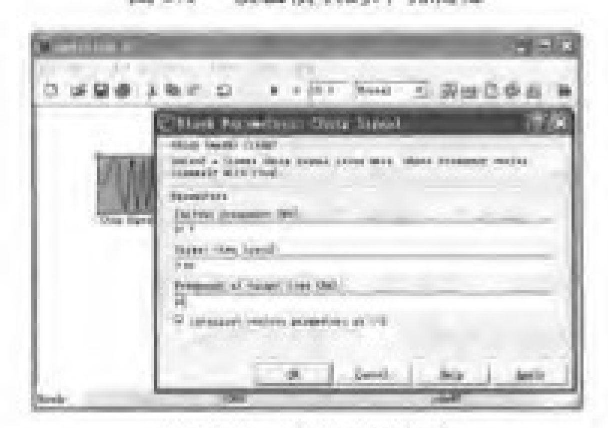


图 8.10　设置模块参数



对于不同的程序模块，MATLAB 显示不同的参数对话框，可以设置各参数的数值。同时，对于不同的模块，MATLAB 都会设置不同的默认参数数值。

step 3 添加"Sine Wave"模块。重复上面的步骤，向模块窗口中添加"Sine Wave"模块，并设置模块的外观属性，得到的结果如图 8.11 所示。



图 8.11　添加"Sine Wave"程序模块

step 4 设置"Sine Wave"模块的属性，查看相应的帮助文件。双击上面步骤中添加的"Sine Wave"模块，在打开的对话框中设置参数，然后单击该对话框中的"Help"按钮，查看关于正弦函数的帮助文件，如图 8.12 所示。



图 8.12 设置模块的属性

step 5 添加数学运算符模块，设置其相应的属性。选择"Simulink Library Browser"对话框左侧的"Math Operations"模块库，在对话框的右侧选择"Add"模块，向模块窗口中添加该模块，并设置其外观属性，得到的结果如图 8.13 所示。



图 8.13 添加数学运算符

step 6 添加显示屏模块，并设置其相应的属性。选择"Simulink Library Browser"对话框左侧的"Sinks"模块库，在对话框的右侧选择"Scope"模块，向模块窗口中添加该模块，并设置其外观属性，得到的结果如图 8.14 所示。

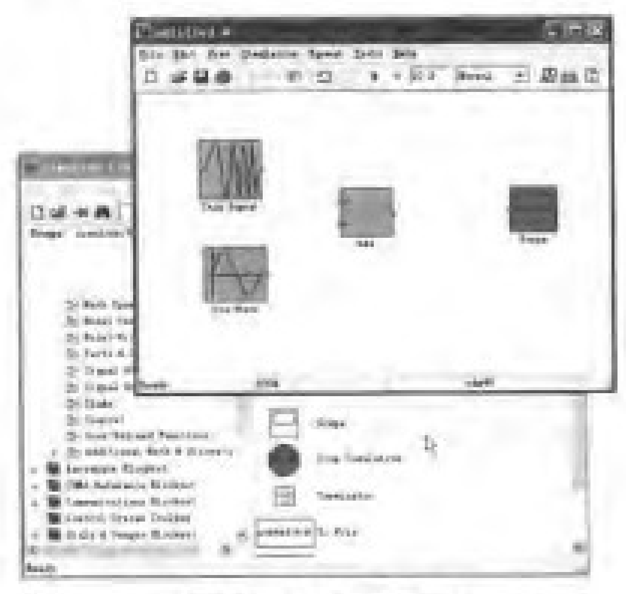


图 8.14　添加显示屏模块

8.1.6 连接模块

延续上面小节的步骤。

step 1 连接程序模块。将鼠标指向“Chirp Singal”模块的右侧输出端，当光标变成十字符时，按住鼠标左键，将其移到“Add”模块的左侧的数步输入端，如图 8.15 所示。

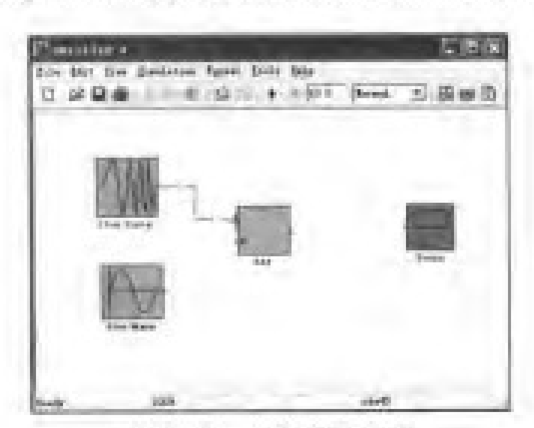


图 8.15　连接程序模块

step 2 连接其他程序模块。可以使用上面步骤中的方法，连接其他的程序模块，然后调整各模块的相对位置，得到的结果如图 8.16 所示。

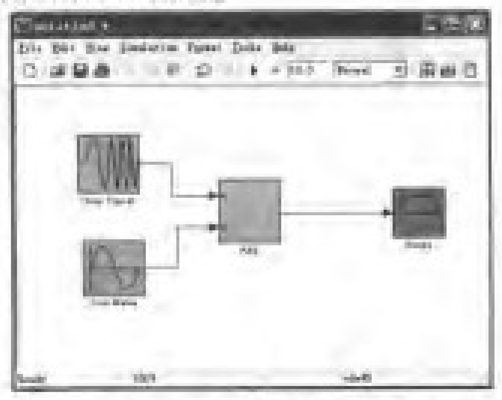


图 8.16　完成连接的程序模块



模块是MATLAB中进行建模的基本单元，可以对其进行复制、移动、旋转等基本操作，修改其对应的外观属性。对于连接线，用户可以产生水平、垂直、斜向的连接线，也可以产生曲折、分支的连接线等。

8.1.7 运行仿真系统

延续上面小节的步骤。

step 1 查看仿真结果。单击模型窗口中的"仿真启动"图标，或者选择编辑栏中的"Simulink"⇨"Start"命令，进行模型的仿真，然后双击上面模块中的"Scope"图标，查看叠加后的信号波形，如图8.17所示。

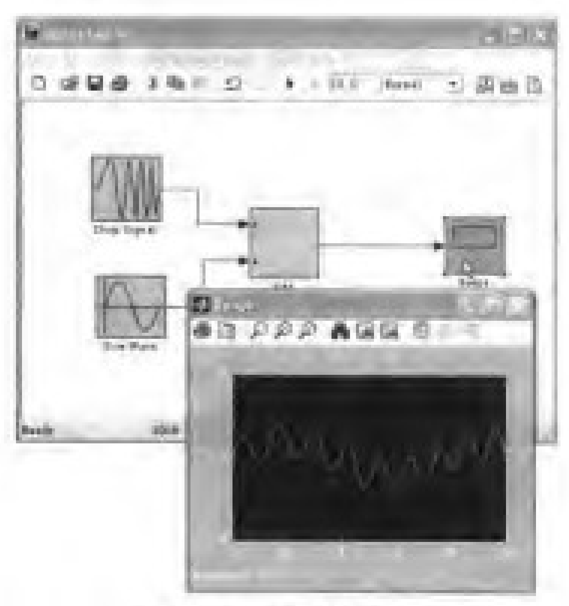


图 8.17 查看仿真结果

step 2 修改仿真显示的结果。单击上面步骤中打开的"Scope"对话框中的"自动刻度"图标，将波形充满整个坐标框，如图8.18所示。



图 8.18 修改仿真显示的结果

step 3 修改仿真的参数。在默认的情况下，模型仿真的时间是10s，用户可以修改该仿真时间，例如，改为20s，重新进行仿真。得到的仿真结果如图8.19所示。

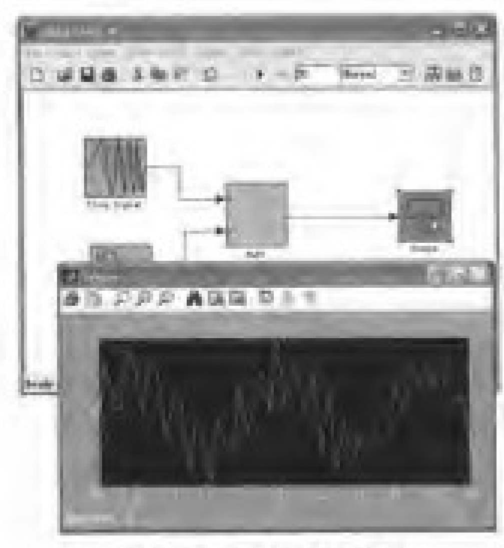


图 8.19　修改仿真参数

8.1.8　模块库浏览器

前面使用了一个简单的实例来说明Simulink的创建方法，本小节将详细介绍创建Simulink的工作环境。当启动Simulink后，将会打开"Simulink Library Browser"对话框，该对话框将是创建Simulink模型的主要环境。下面详细介绍该工作环境的主要功能。

首先，启动Simulink，打开"Simulink Library Browser"对话框，如图8.20所示。



图 8.20　"Simulink Library Browser"对话框

可以使用工具栏中的按钮对Simulink模型进行设置，对应的工具栏如图8.21所示。



图 8.21　对话框的工具栏

将上面工具栏中的按钮依次命名为1－4，下面详细介绍工具栏对应按钮的功能。

- **1号按钮**：标准的Windows工具菜单，表示新建一个Simulink模型。
- **2号按钮**：标准的Windows工具菜单，表示打开一个已经创建的Simulink模型。
- **3号按钮**：将Simulink的"Simulink Library Browser"对话框设置在桌面的最高层。

✦ **4号按钮：**进行关键词查找。可以在其后面的关键词填写框中输入模块的关键词，MATLAB将会在对应的路径中进行查找。

由于模块查找是比较重要的使用方法，下面以查找包含关键词 "clock" 的模块为例，演示如何使用该项功能，如图 8.22 所示。



图 8.22　模块搜索

当在关键词填写栏中输入 "clock" 之后，MATLAB 便在指定的模型库中进行查找，当查找到模块之后，将该模块醒目显示，并给出对应的说明。如果查找的结果多于 1 个，可以连续单击 "查找" 按钮，依次循环查看查找的多个结果。



如果在指定的模块库中没有找到包含关键词的模块，MATLAB会在 "Simulink Library Browser" 对话框中的文字说明区中加以说明。

除了工具栏之外，"Simulink Library Browser" 对话框中的模型库总览表和子目录对用户创建Simulink模型也是十分重要的使用内容。其中，模块库总览表位于对话框中的左侧，具有很好的分层结构。其中第一层是 Simulink 模块组，本书中包括 Simulink、Aerospace Blockset、Communication Blockset、Control System Toolbox 等多个模块组，该模块组中包含的模块个数取决于用户安装的组件；第二层内容包含的是子模块库，以 SimMechanics 模块组为例，其中包含 Bodies、Constraints& Drivers、Force Elements、Joints、Sensors &Actuators 和 Utilities 等 6 个子模块库。



如果在第二层的子模块库中，对应的模块前面还有 "+" 标记，表明在该子模块库中，还包含了下一层的子模块库内容。

8.1.9　Simulink 模型窗口界面

在 Simulink 中，除了 "Simulink Library Browser" 对话框之外，还提供空白模型窗口，可以在该窗口界面中对 Simulink 进行必要的操作。在本小节中，以 MATLAB7.0 提供的 "f14.mdl" 文件为例，介绍模型窗口的使用方法。首先，需要打开 demo 文件夹下的 "f14.mdl" 文件，如图 8.23 所示。

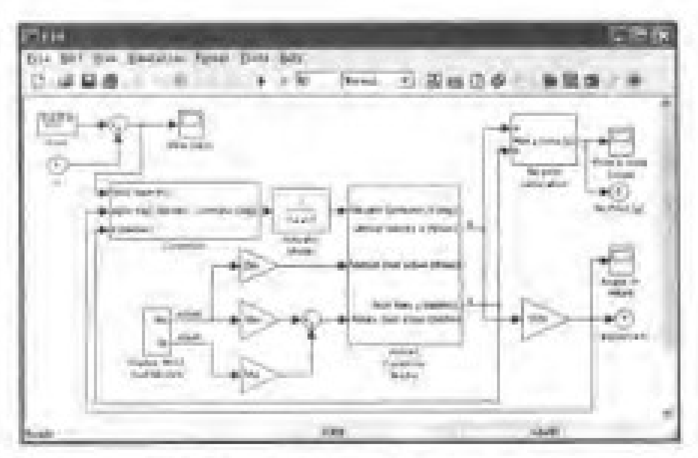


图 8.23　Simulink 模型窗口界面

上面的对话框只是"模型框"的单窗口形式，可以单击"Toggle Model Browser"按钮，切换到双窗口形式。在该双窗口形式中，左侧窗口是"Model Browser"，显示该模型的分层子系统目录，右侧窗口显示相应系统的连接图，如图 8.24 所示。

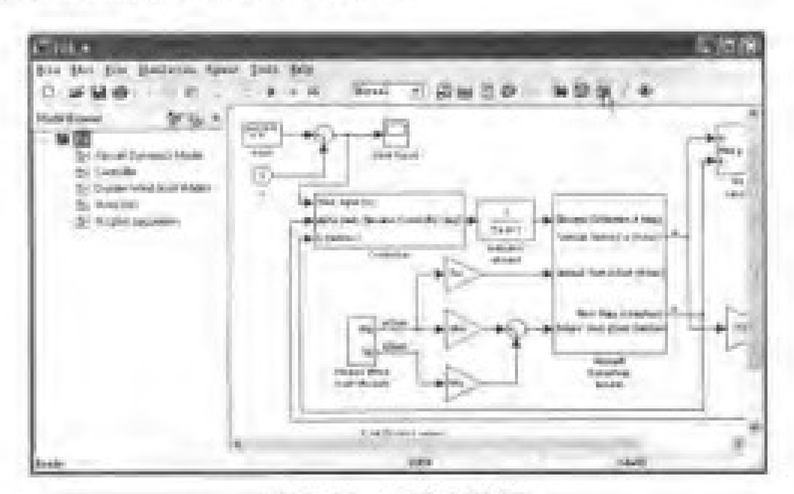


图 8.24　双窗口形式

根据上面两个简单的对话框，Simulink 中的模型窗的基础组成部分是：菜单栏、工具栏、编辑框和状态栏，下面将详细介绍这些组成部分的功能。

其中，模型窗的工具栏如图 8.25 所示。



图 8.25　模型窗的工具栏

将工具栏中的按钮从左到右依次命名为 1、2、3……按钮，文字框 1，下拉菜单 1 等，下面依次介绍各自的功能。

- **按钮 1～7**：Windows 的标准按钮，对应的功能就是 Windows 操作系统之下的功能，依次为新建、打开、保存、打印、剪切、复制和粘贴模块。
- **按钮 8**：撤销上一个操作步骤。
- **按钮 9**：返回下一个操作步骤。
- **按钮 10**：开始仿真按钮，单击该按钮，就可以开始整个系统的仿真。
- **按钮 11**：停止仿真。单击该按钮，就可以停止整个系统的仿真。

- **文字框 1**：设置仿真时间。默认情况下，该数值为 10.0，用户可以输入合适的仿真时间。
- **下拉菜单 1**：设置仿真的加速模式。MATLAB 提供 Normal、Accelerator 和 External 三种仿真加速模式，用户可以根据需要选择对应的加速模式。
- **按钮 12**：准备系统的仿真。
- **按钮 13**：产生 RTW 程序代码。
- **按钮 14**：刷新整个系统。
- **按钮 15**：更新整个系统。
- **按钮 16**：为子系统产生程序代码。
- **按钮 17**：显示 Simulink 的模块库浏览器。
- **按钮 18**：打开模块管理器。
- **按钮 19**：打开或者隐藏模型浏览器。
- **按钮 20**：转到上一级系统中。
- **按钮 21**：打开调试器。



上面工具栏中的各个按钮并不会在同一个时间内有效。对于不同的 Simulink 对象，MATLAB 会选用不同的工具栏按钮。

在模型窗口的状态栏中，会显示整个仿真系统的状态，以上面步骤中选用的实例为例，从左到右的文字的含义依次如下。

- **Ready**：表示模型已经准备就绪，等待系统的仿真命令。
- **100%**：表示编辑框的显示比例。
- **ode45**：表示仿真选用的积分算法是"ode45"。

除了上面的参数之外，当在进行模型仿真的时候，在状态栏的其他空白格中还会出现对应的动态仿真信息。

在 MATLAB 的模型窗口界面中，多数主要的功能都是通过该界面对应的菜单选项来实现的，下面将详细介绍各菜单选项的主要功能。为了节省本章的篇幅，在介绍菜单选项功能的时候，将不介绍 Windows 标准菜单的功能。

8.1.10 "File"菜单

"File"菜单的主要选项以及对应的功能如表 8.1 所示。

表 8.1 "File"菜单选项和功能

主要的菜单选项	功能
Model Properties	设置 Simulink 的模型属性
Preferences	设置 Simulink 界面形态的默认属性
Source Control	设置 Simulink 和 SCS 的接口
Print	打印模型
Print Details	生成HTML格式的模型报告文件，包括模型参数的设置等主要信息

对于其他子菜单的功能，在本小节中就不详细介绍了，只介绍生成模型报告文件的方法。还以上面的实例为例，可以选择菜单栏中的"File" ⇨"Print Datails"命令，打开对应的"Print Datails-f14"对话框。可以在该对话框中设定相应的报告参数，然后单击该对话框中的"Print"按钮，如图

8.26 所示。

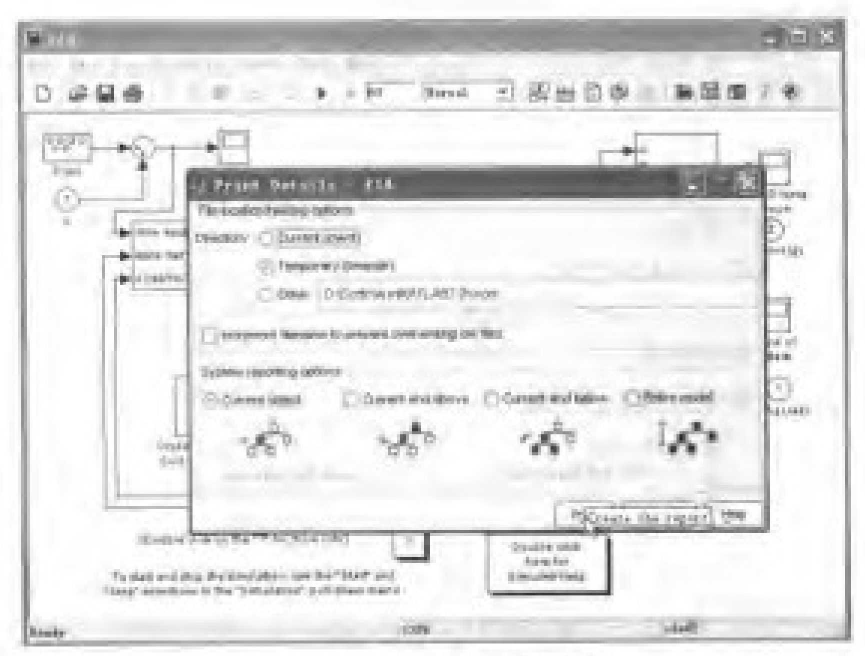


图 8.26　生成模型报告文件

8.1.11　"Edit" 菜单

"Edit" 菜单的主要选项以及对应的功能如表 8.2 所示。

表 8.2　"Edit" 菜单选项和功能

主要的菜单选项	功能
Find	搜索 Simulink 系统内的模型块、信号、声明等各种对象
Block Properties	打开模块属性对话框
Create Subsystem	创建子系统
Mask Subsystem	封装子系统
Look under Mask	查看精装子系统中的内部结构
Signal Properties	设置信号属性
Edit Mask	编辑封装子系统
Subsystem Parameters	设置子系统的参数
Mask Parameters	封装好的子系统的参数设置
Update Diagram	更新模型框图的外观属性



在上面的菜单选项中，当涉及子系统的操作时，必须选中对应的子系统才能激活该选项，例如 "Mask Subsystem" 选项；当涉及模块的操作时，必须选中对应的模块才能激活对应的选项，例如 "Block Properties" 选项。

由于上面大部分的选项菜单都必须选中各自的对象，因此在本小节中还是以上面的实例为例，介绍 "Find" 菜单的对应使用方法。在 Simulink 的模型窗口中，当选择 "Find" 菜单选项后，打开对应的 "Find" 对话框，在该对话框中可以搜索各种常见的对象。选择菜单栏中的 "Edit" ⇨ "Find" 命令，打开 "Find: f14" 对话框，如图 8.27 所示。



图 8.27　搜索对话框

在上面的对话框中，可以在 "Filter options" 面板设置搜索的模式，然后在 "Search criteria" 面板中输入搜索的关键词。如果搜索的系统包含多个系统或者子系统，则需要在 "Start in system" 下拉菜单中选择搜索范围。最后，在 "Match case" 选项框中选择搜索匹配方式，单击对话框中的 "Find" 按钮，进行搜索，如图 8.28 所示。



图 8.28　搜索结果

从上面的搜索结果中，取消选中 "Stateflow objects" 选项，同时选择 "Match whole word" 的 Match 方式，选中 "Look inside masked systems" 选框，在上面的搜索方式下关键词 "Gain" 的搜索结果只有两个结果。

保持上面的关键词不变，修改各个搜索参数，可以查看新的搜索结果，如图 8.29 所示。

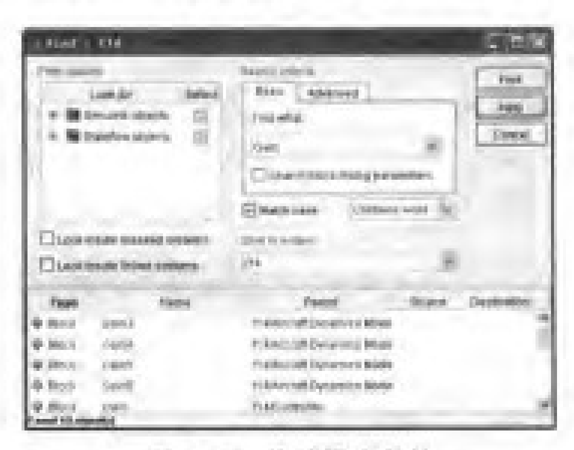


图 8.29　修改搜索条件

从上面的搜索结果可以看出，保持 "Gain" 关键词不变，修改搜索条件后，得到的搜索条件就会发生很大的变化，得到的搜索结果有 13 个。

8.1.12 "View" 菜单

"View" 菜单的主要选项以及对应的功能如表 8.3 所示。

表 8.3 "View" 菜单选项和功能

主要的菜单选项	功能
Block Data Tips Options	设定当鼠标移到某个模块时，是否显示提示信息
Library Browser	打开模块库浏览器
Port Values	设置如果通过鼠标操作，显示模块端口的当前值
Model Explorer	打开模块资源管理器，进行模块资源管理

在模块资源管理器中，查看模块的参数设置、仿真参数设置、解法器选择、模块的各种信息等各种信息。选择菜单选项 "View" ⇒ "Model Explorer" 命令，打开 "Model Explorer" 对话框，如图 8.30 所示。

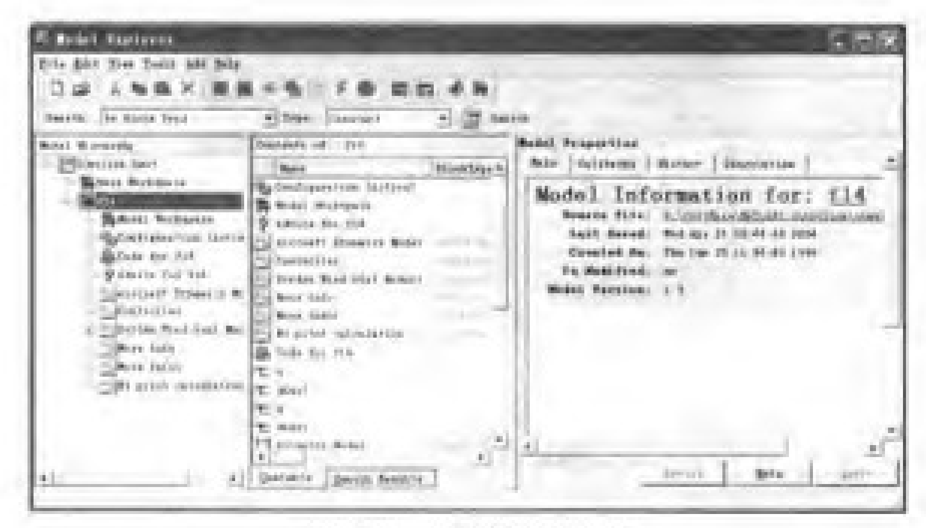


图 8.30 模型资源管理器

在上面的对话框中，包含 "Model Hierarchy"、"Contents" 和 "Dialog" 三个主要面板，同时包含一个主要工具栏和搜索工具栏。在 "Model Hierarchy" 面板中，主要显示 Simulink 模块的树形体系结构，主要体系结构的节点包括 Simulink Root、Base Workspace 和 Model Nodes 等，可以在该面板中查看对应对象的各种属性。

在 "Contents" 面板中将用列表的形式显示用户在 "Model Hierarchy" 面板中选择的对象的属性，或者显示用户在搜索工具栏中查询的结果。例如，在本例中可以选择 "Aircraft Dynamics Model" 选项，查看对应的属性内容，如图 8.31 所示。



图 8.31 查看对象的属性内容



在"Contents"面板中显示的对象属性内容取决于用户在"Model Hierarchy"面板中选择的对象类型。如果选择模型或者子系统，在默认情况下，"Contents"面板中将显示该模型或者子系统所包含的高层模块的名称或者类型。

除了上面显示的默认"Contents"面板之外，MATLAB 还提供自定义内容面板，用户可以使用自定义面板来设置显示属性的内容选项。选择"Model Explorer"对话框中的"View" ⇨ "Customize Contents"命令，在默认的显示面板中添加"Customize Contents"面板，可以在其中选择显示的属性内容，如图 8.32 所示。



图 8.32 显示"Customize Contents"面板

在上面的对话框中，选择只显示各个对象的 Name、MaskType、TreatAsAutomaticUnit 和 Priority 属性，因此在右侧的"Contents"面板中只显示这些属性数值。



关于该对话框中的主要工具栏和搜索工具栏的使用方法，将在后面的章节中对应的部分详细介绍。限于篇幅，这里就不详细展开了。

8.1.13 "Simulation"菜单

和前面的菜单选项相比，Simulation 菜单所包含的选项比较简单。Start 菜单选项表示开始运行仿真系统，Stop 菜单选项表示停止仿真系统，"Configuration Parameters"菜单选项则表示设置仿真参数和选择解法器。选择"Simulation" ⇨ "Configuration Parameters"命令，打开"Configuration Parameters: f14/ Configuration"对话框，如图 8.33 所示。

在上面的仿真参数对话框中，主要包含 Slover、Data Import/Export、Optimization、Diagnostics、Hardware Implementation 和 Model Referencing 等面板，可以在对应的面板中设置仿真的各种参数数值。



限于篇幅，关于上面各个属性面板的使用方法在本小节中就不详细展开了。这些内容将在后面章节中对应的部分详细介绍。



图 8.33 仿真参数对话框

8.1.14 "Help" 菜单

"Help" 菜单的主要选项以及对应的功能如表 8.4 所示。

表 8.4 "Help" 菜单选项和功能

主要的菜单选项	功能
Using Simulink	显示关于 Simulink 的帮助部分
Blocks	显示按字母排列的 Blocks 帮助部分
Block Suport Table	显示模型所支持的数据类型的帮助内容
About Simulink	显示 Simulink 的版本信息

选择 "Help" ⇨ "Block Suport Table" 命令，查看模块支持的数据类型，如图 8.34 所示。

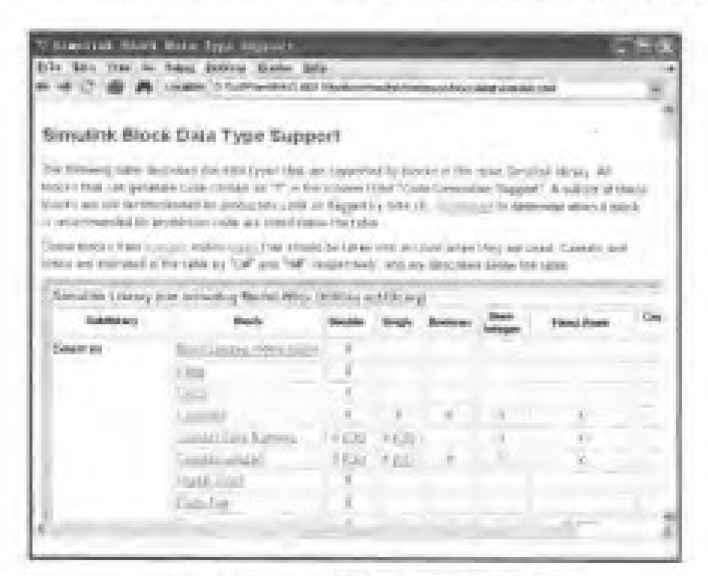


图 8.34 模型所支持的数据类型



关于 Format 和 Tools 菜单的选项，由于内容比较简单，这里就不详细介绍了，请用户自行查看相应的帮助文件。

8.2 Simulink 的数据类型

熟悉编程语言的读者对“数据类型”的概念应该不会陌生。数据类型直接决定系统分配给不同变量的存储空间，也决定程序运算的精度范围和性能等。MATLAB 也是一种程序语言，也需要设置 Simulink 中信号和模块参数的数据类型。

实质上，Simulink在进行整个系统的仿真之前和仿真过程中都会进行一个额外的数据类型检查。这种检查是系统自行完成的，用户无须为此编写专门的程序代码。这种数据类型的检查也可以认为是模型的安全性检测，可以确保整个模型（或者被称为系统）产生的程序代码不会出现上溢或者下溢，不至于产生一些不精确的程序结果。在 Simulink 中默认的数据类型是 Double，如果没有特别指出模型的数据类型，一般都将是 Double 数据类型。

8.2.1 Simulink 支持的数据类型

作为 MATLAB 的重要组件，Simulink 支持 MATLAB 中所有的内置数据类型，所谓的内置数据类型是指 MATLAB 自行定义的数据类型。

为了帮助读者理解 Simulink 支持的数据类型，表 8.5 列出了内置数据类型。

表 8.5　MATLAB 的内置数据类型

数据类型	说明
Double	双精度浮点类型
Single	单精度浮点类型
Int8/Uint8	有符号 / 无符号的 8 位整数
Int16/Uint16	有符号 / 无符号的 16 位整数
Int32/Uint32	有符号 / 无符号的 32 位整数

除了上表中列出的内置数据类型，Simulink 还专门定义了 Boolean 数据类型，取值为 0 和 1，其内部数据类型为 Uint8（无符号的8位整数）。绝大多数的Simulink模型的数据类型都是默认的Double类型，有些模块需要 Boolean 数据类型，还有一些模块支持多数据的输入或者复数信号等。

对于一个已经完成的Simulink模块，可以使用多种方法来查看模块或者信号所支持的数据类型。在一个完整的 Simulink 的仿真系统中，如果希望了解某单个模块所支持的数据类型，可以双击该模块，打开对应的模块参数设置对话框，在其中查看数据类型。还是以“f14.mdl”文件为例，双击其中的“Nz Pilot”模块，在打开的模块参数对话框中选择“Singal specification”选项卡，如图 8.35 所示。



如果在打开的参数对话框中没有特别指出该模块支持的数据类型，表明该模块只支持默认的 Double 数据类型。

如果希望一次查看多个模块的输入 / 输出数据类型，可以选择菜单栏中的“Format”⇨“Prot/Signal Displays”⇨“Prot Data Types”命令，查看各模块的输出 / 输入的数据类型，得到的结果如图 8.36 所示。

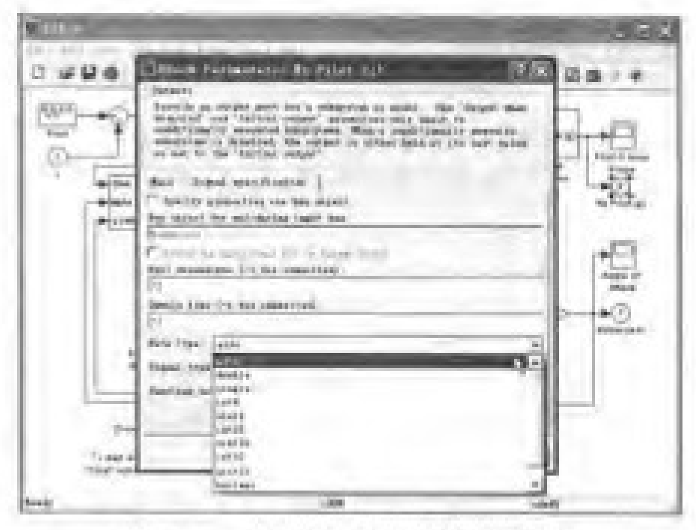


图 8.35 查看模块所支持的数据类型

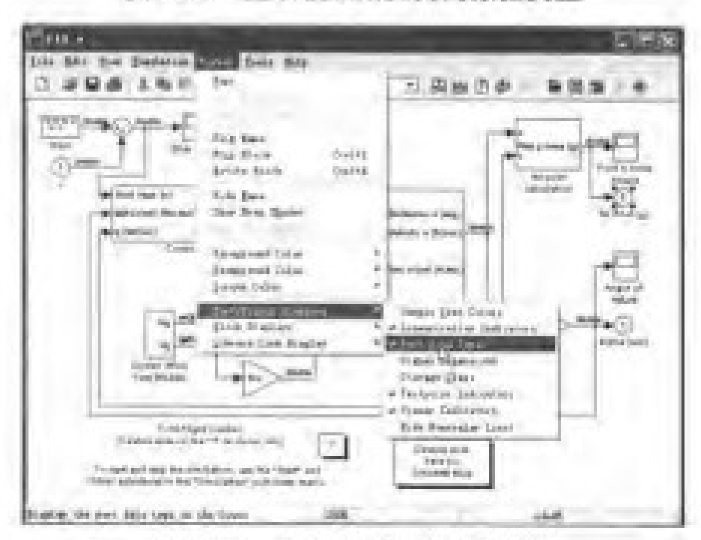


图 8.36 查看各模块的数据类型

当选择对应的菜单选项后，会在模型窗口中各个模块的输入口和输出口中显示参数的数据类型。除了可以查看各个对象的数据类型之外，还可以为各个对象设置数据类型，输入的格式为 Type (value)。还是以上面小节的实例为例，可以修改模块中对应的输入数值，如图 8.37 所示。

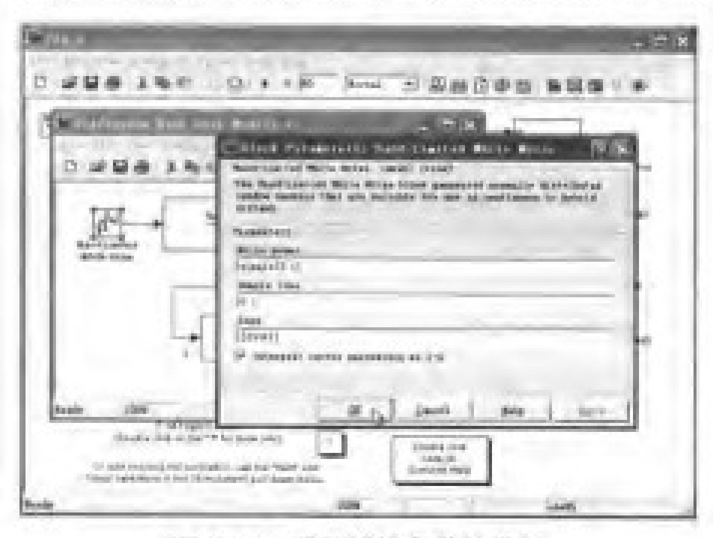


图 8.37 设置模块参数的数值

在上面的对话框中，在对应的文本框中输入"Single(5.1)"，将模块的参数设置为 5.1 的单精度浮点数据类型。如果模块不支持用户设定的数据类型，那么 MATLAB 会弹出相应的警告信息。

8.2.2 Simulink 中的数据传递

一般情况下，Simulink 系统包含多个数据模块和信号模块，而且在仿真过程中，各个不同的模块之间会不断地实现数据传递，但是这些不同类型的模块所支持的数据类型往往不完全相同。如果使用信号线相连的两个模块所支持的数据类型有冲突（这里所说的数据类型是指信号模块中的输入/输出数据类型，不是指模块参数的数据类型），当进行模块仿真的时候，MATLAB 在查看端口数据类型或者更新数据类型时会弹出提示对话框，提示用户出现冲突的信号和端口，同时，这些有冲突的信号路径会高亮显示。

实质上，在实际建模过程中输入信号的数据类型和模块的数据类型往往是不同的，Simulink 在计算过程中会将参数类型转换为信号的数据类型。并不是所有的情况都会自动转换，当信号的数据类型无法表示参数数值时，将会自动中断仿真，并给出错误信息。

8.2.3 Simulink 中的数据转换实例

例 8.2 使用简单的数学算例演示 Simulink 的数据类型转换规则。

step 1 打开 Simulink 的空白模板编辑器，向其中添加"Constant"数值模块，将其数值设置为 2.5，同时打开其参数管理器，将其数据类型选择为"Boolean"，如图 8.38 所示。

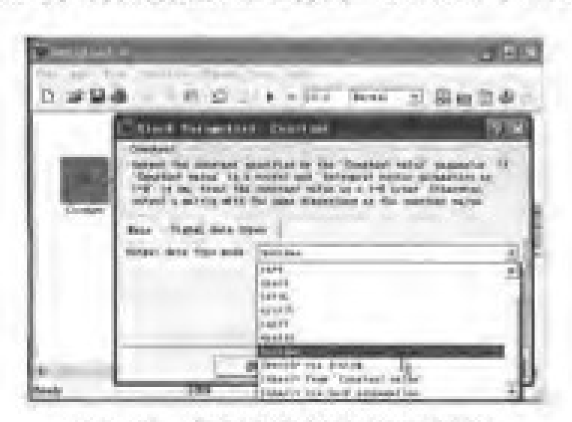


图 8.38 添加参数模块并设置属性

step 2 添加下一个常数数值模块，将其数值设置为 1.5，然后将其输出的数据类型设置为"Boolean"，得到的结果如图 8.39 所示。

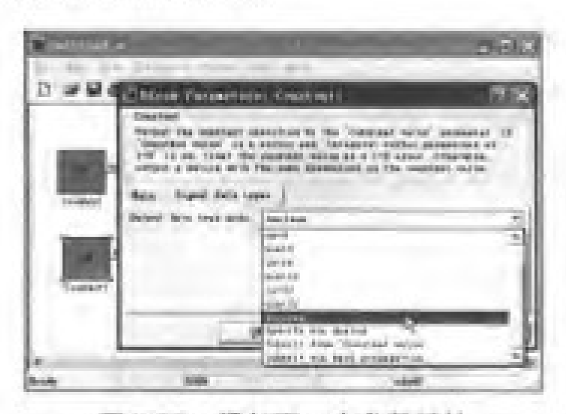


图 8.39 添加下一个参数模块



为了使模块在外观上更加美观，将上面两个常数数值模块的背景颜色设置为粉红色。设置的方法在前面介绍过，这里就不重复介绍了。

step 3 添加运算模块。为了演示上面模块中的数学运算过程，向上面的模块中添加加法、积分运算模块，然后添加“Scope”模块，如图8.40所示。

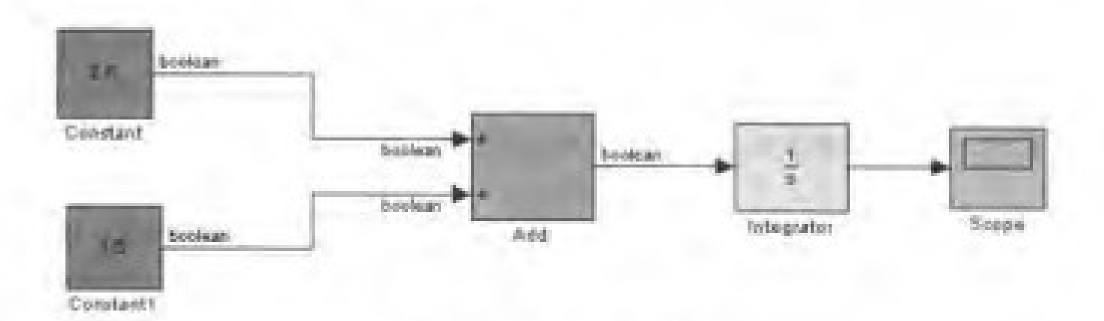


图8.40　添加的模块系统

step 4 进行系统仿真。单击空白模块界面中的“开始仿真”按钮，将上面的系统进行仿真，得到的提示对话框如图8.41所示。

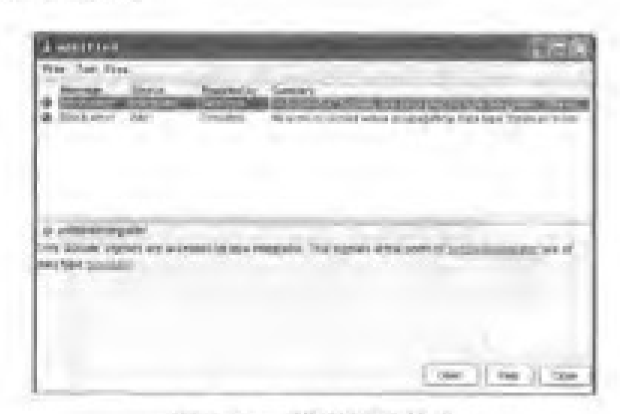


图8.41　错误提示信息

在上面的对话框中，系统提示信息“Only 'double' signals are accepted by type Integrator. The signals at the ports of 'untitled/Integrator' are of data type 'boolean'”，表示“Integrator”模块只接收“double”数据类型，而在本仿真系统中，“Integrator”模块输入数据的数据类型是“Boolean”，Simulink无法在这两种数据类型中实现转换。因此提示发生了数据类型的错误，并在模块界面中高亮显示错误的“Integrator”模块，如图8.42所示。

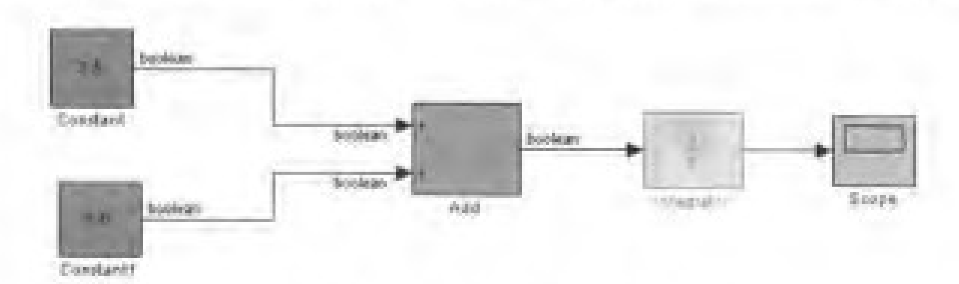


图8.42　高亮显示错误模块

step 5 修改仿真系统。为了修改上面的仿真系统，需要在“常数”模块和“加法”模块中间添加“Commonly Used Blocks”模块库下的“Data Type Conversion”模块，实现数据转换，并设置该数据模块的属性，如图 8.43 所示。

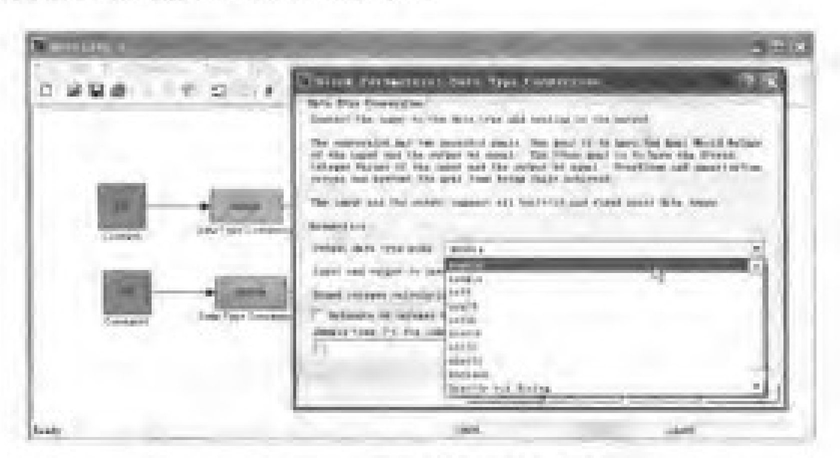


图 8.43 添加数据转换模块

在上面步骤中打开的“Data Type Conversion”模块参数设置对话框中，将模块的输出数据类型设置为“double”。这样，就可以将 Boolean 数据类型转换为 Double 数据类型，然后将这些数据进行叠加，最后进行积分。



说明 在 Simulink 中，“Data Type Conversion”模块的主要功能是将输入数据的数据类型转换为指定的数据类型，但是数据转换并不能保证原始数据不变，因此使用这个模块时需要谨慎。

step 6 查看仿真结果。在修改仿真模块之后，就可以进行系统仿真，查看仿真结果，如图 8.44 所示。

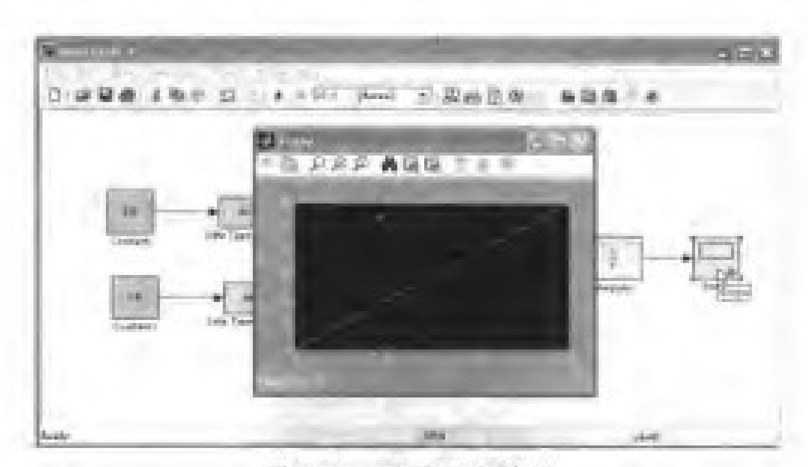


图 8.44 查看仿真结果



说明 从上面的仿真结果中可以看出，积分结果是“y=2x”，而不是“y=4x”。这就表明尽管使用了“Data Type Conversion”模块，但是转换结果是 1 或者 0，而不是原始的常数模块中的数值 2.5 和 1.5，两个模块转换的数值结果都是 1。

8.2.4 向量化模块

在Simulink模块库中几乎所有模块都是向量化的模块（Blocks Vectorized），向量化模块大输入量和输出量之间的关系是符合数学规则的向量关系的。向量信号是多个信号的集合，对应着模块窗口中几条并行连线的合成。在默认情况下，Simulink中大多数的模块输出数据是标量信号，当输入的是向量信号而模块的参数类型是标量时，Simulink会进行自动匹配。

例 8.3 使信号模块输出向量信号。

step 1 打开新的空白模型界面，在其中添加"Chirp Signal"信号，然后双击该信号模块，打开对应的参数设置器，在其中设置信号的参数，如图8.45所示。

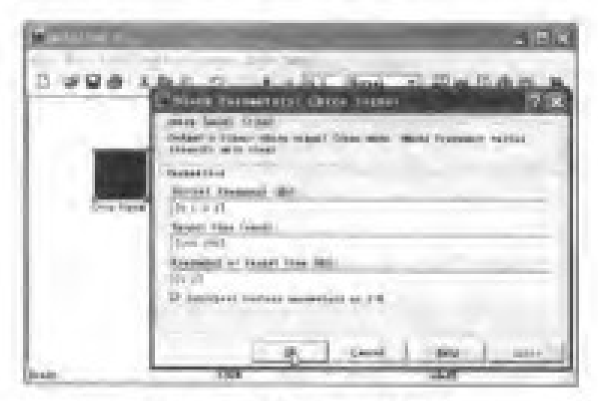


图 8.45 设置模块的属性

在上面的对话框中，在"Initial frequency"文本框中输入向量"[0.1 0.4]"，在"Target time"文本框中输入向量"[100 250]"，在"Frequency at target time"文本框中输入向量"[1 2]"，然后单击对话框中的"OK"按钮，完成向量的输入。



在Simulink中，"Chirp Signal"模块的输出数是一个线性信号，该信号是一个频率随着时间线性变化的正弦波形。

step 2 向上面的模块中添加"Scope"模块，运行仿真，再双击"Scope"模块，查看仿真结果，得到的结果如图8.46所示。

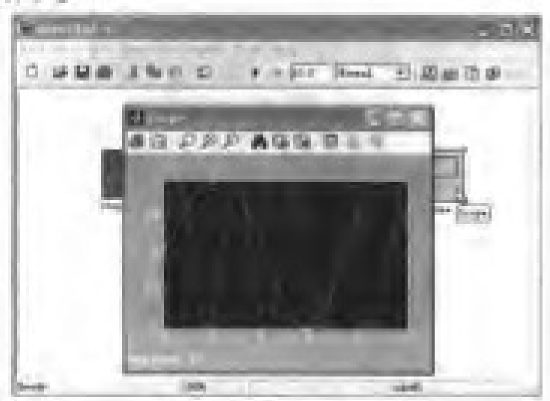


图 8.46 查看仿真结果



在上面的实例中，模块中所有参数都是按照向量元素的位置和信号对应的，不存在向量的匹配问题，因此在输入向量参数的时候需要自行注意这种匹配关系。

8.2.5 使用 Mux 模块

例 8.4 使用 "Mux" 模块输出向量信号。

step 1 打开新的空白模型界面，在其中添加 "Chirp Signal" 信号，然后双击该信号模块，打开对应的参数设置器，在其中设置信号的参数，如图 8.47 所示。

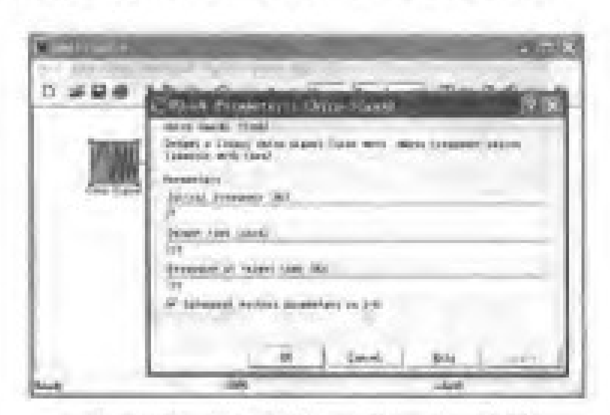


图 8.47 添加 "Chirp Signal" 信号

step 2 在其中添加 "Sine Wave" 信号，双击该信号模块，打开对应的参数设置器，在其中设置信号的参数，如图 8.48 所示。



图 8.48 添加 "Sine Wave" 信号

step 3 添加 "Mux" 模块，将上面两个信号模块进行向量化处理，然后添加 "Scope" 模块，显示向量化处理的结果，如图 8.49 所示。

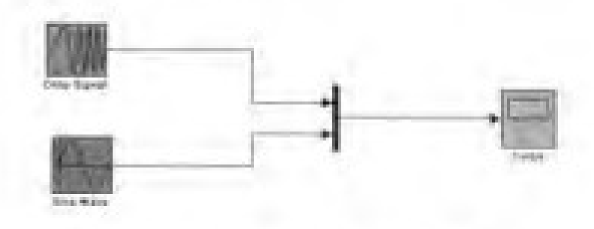


图 8.49 添加 "Mux" 模块

在Simulink中，Mux模块的功能是将输入参量转换为一个单个输出数据。该模块接收的输入参量包括标量、向量或者矩阵信号，根据输入参量的不同数据类型，其输出数据结果可能是一个向量或者复合信号（例如，包括矩阵和向量的复合信号）。

step 4 设置"Mux"模块的属性。在MATLAB中，可以设置"Mux"模块的属性包括输入数据的数目和显示的数据类型，双击上面系统中的"Mux"模块，得到的参数对话框如图8.50所示。



图8.50 设置"Mux"模块的属性

step 5 查看仿真的结果。单击模块界面中的"开始仿真"按钮，双击上面系统中的"Scope"模块，查看仿真的结果，得到的结果如图8.51所示。

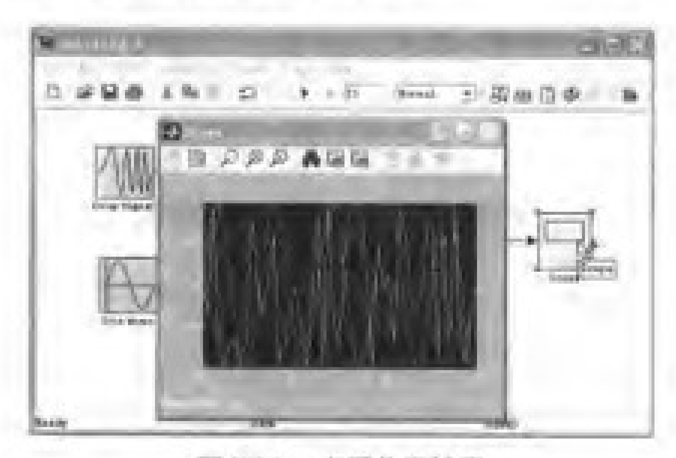


图8.51 查看仿真结果

从上面的实例中可以看出，这里使用了Chirp波 $w(t)$和正弦波 $s(t)$两个标量输入信号，经过"复用"模块的处理，得到的结果是一个向量波形 $v(t)=\begin{bmatrix} w(t) \\ s(t) \end{bmatrix}$。



之所以在上面两个步骤中分别设置两个波形的频率属性，是为了在最后的步骤中将两个波形叉开显示，使得结果更加明显。

8.2.6 标量扩展

除了前面提到的内容外，MATLAB还提供标量扩展（Scalar expansion）功能。该功能具体就是向量化模块执行符合规则运算所必须具备的自适应能力。标量扩展包含的内容为输入的标量扩展和参数标量扩展，下面举例详细说明这些用法。

例8.5 通过"乘/除"模块来显示输入扩展功能。

step 1 打开新的空白模型界面，在其中添加"Signal Generator"和"Constant"模块，然后双击

"Signal Generator" 模块，打开参数设置器，设置信号的参数，如图 8.52 所示。

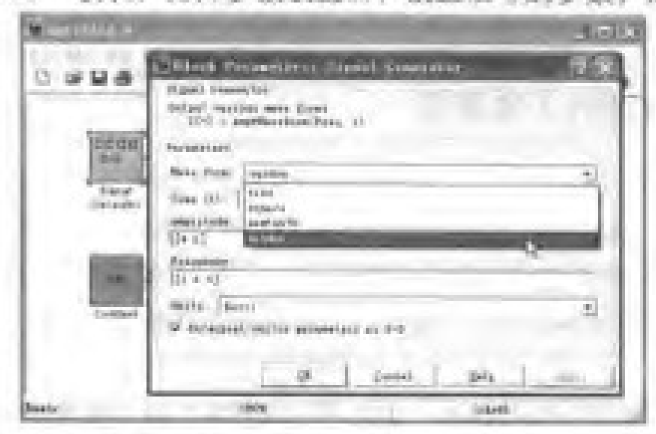


图 8.52 添加 "Signal Generator" 模块

step 2 添加 "Divide" 模块和 "Scope" 模块。在上面的模块系统中添加 "Divide" 模块和 "Scope" 模块，如图 8.53 所示。

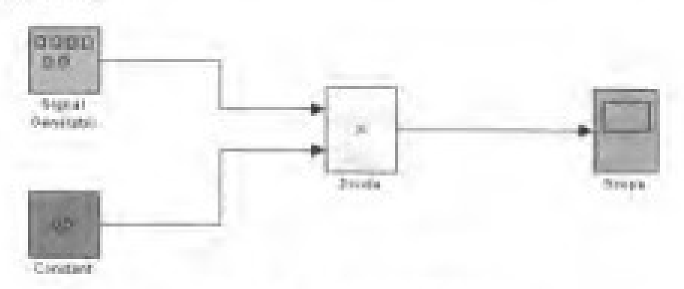


图 8.53 添加程序模块

step 3 查看仿真的结果。单击模块界面中的 "开始仿真" 按钮，然后双击系统中的 "Scope" 模块，查看仿真的结果，得到的结果如图 8.54 所示。

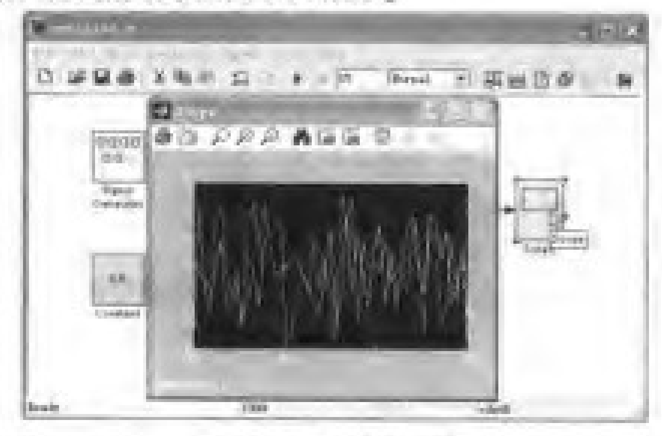


图 8.54 查看仿真结果



在上面的实例中，"Signal Generator" 模块产生了一个向量信号，然后使用乘法模块将向量信号结果同时乘以 0.5，得到最后的信号结果。

8.3 Simulink 的基本操作

前面介绍了关于 Simulink 的操作界面和数据类型，从本节开始，将开始介绍如何创建 Simulink

模型。由于Simulink可以创建各种复杂的系统，本节将主要介绍创建Simulink模型中共用的知识和操作。

8.3.1 Simulink 模型的工作原理

尽管Simulink系统屏蔽了许多烦琐的编程工作，似乎用户不需要了解Simulink的基础工作原理，但是，如果深入了解关于Simulink的基础原理，就可以更加高效灵活地使用Simulink模型。创建Simulink模型的基本步骤包括创建模型图标和控制系统仿真，但是读者对模型的图标之间的映射关系以及Simulink如何利用这种映射关系进行仿真，可能还是比较陌生的，在本小节中将简要介绍这些基本内容。

一般来讲，系统或者模型的当前数值，或者产出都是某个临时变量原来数值的函数，这些临时变量就是所谓的"状态"数值。当Simulink从某个模型的图形化模块系统中计算结果的时候，首先需要保存当前时间步骤中变量的数值，然后以此为基础计算变量后续数值。Simulink通过模拟定义状态的模块来完成上面的步骤。其中，系统状态随时间而变化的函数关系（也可以称为数学模型）是由一系列数学方程式来描述的，每一个模块都代表一组数学方程组，Simulink称这些方程组为模型的方法。

图8.55代表定义状态的系统模块的图形原理。

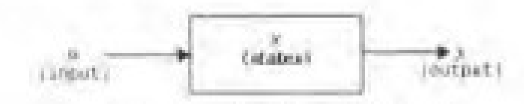


图8.55　模块的图形化形式

从宏观上来看，Simulink模型通常包括三种组件：信号源、系统和信号接收器（或者被称为信宿）。其中信号源可以是常数、正弦波、阶梯波等，信号接收器则可以是示波器或者图形记录仪等。这三种组件可以直接从Simulink的模型库中直接获取，也可以使用提供的模型自行创建系统。



并不是所有的Simulink模型都包括上面的三个组件，用于研究初始条件对系统影响的Simulink模型就不必包含信号源组件。

Simulink对模型系统进行仿真的过程，也就是在用户定义的时间段内根据模型提供的信息计算系统的状态和输出的过程。大致来讲，Simulink的仿真过程可以分为模型编译阶段、连接阶段和仿真环阶段等三个阶段，每个阶段都会完成该阶段的任务。

其中，模型编译阶段的主要任务包括调用模型编译器，将模型编译成为可以执行的形式，详细的任务列表如下：

- 评价模型参数的表达式并确定它们的数值；
- 确定信号的属性；
- 传递信号属性并确定没有定义的信号属性；
- 对系统模块进行优化；
- 展开模型的继承关系；
- 确定模型运行的优先级；
- 确定模块的采样时间。

连接阶段的主要任务包括创建按照执行的次序安排的方法运行列表，同时定位和初始化存储在

每个模块的信息。

仿真环阶段主要分为两个部分，第一个阶段是仿真环初始化阶段，这个阶段只运行一次，主要用于初始化系统的状态和输出；第二个阶段是仿真环迭代阶段，该阶段在定义的时间段内每隔一个时间步就重复运行一次，用于在每一个时间步计算模型的新输入、输出和状态。

8.3.2 模块的操作

模块是创建Simulink模型的基础元素，用户可以合适的方式把各种不同的模块连接在一起就可以创建各种动态系统。因此，有必要熟悉模块的操作方法，才能灵活高效地创建各种仿真系统。

在详细介绍如何操作模块之前，首先提示一个使用技巧。为了更加了解模块的属性和特征，可以设置属性。当使用鼠标在模块上方移动时，Simulink可以显示关于该模块的基本属性信息。至于需要显示哪些属性，可以通过模块界面中的"View"菜单选项下的"Block data tips"菜单选项来设置，下面举例详细说明。

例 8.6 设置模块显示的属性选项。

step 1 打开前面保存的"Sim4"文件，然后选择菜单栏中的"View" ⇨ "Block Data Tips Options" ⇨ "Parameter Names And Values"命令，如图 8.56 所示。

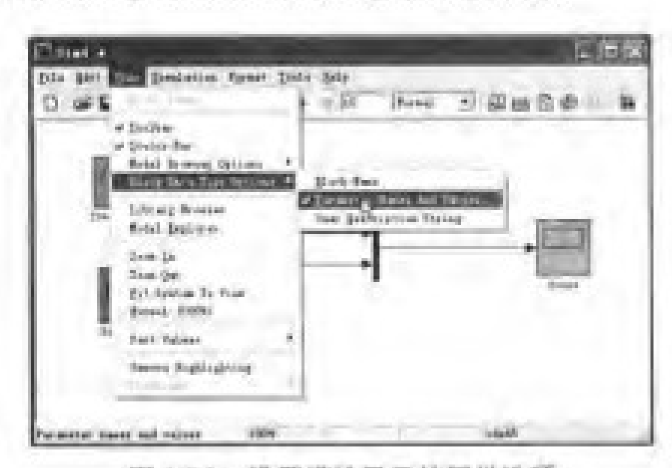


图 8.56 设置模块显示的属性选项

step 2 查看模块的属性。完成上面的设置之后，将鼠标移到"Sine Wave"模块，系统显示对应的属性和数值，如图 8.57 所示。

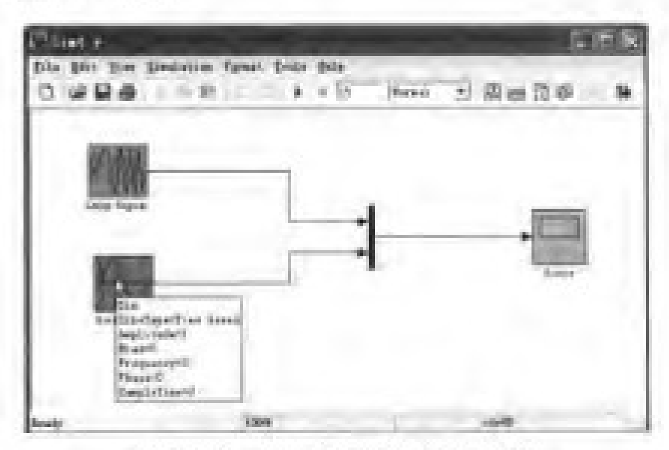


图 8.57 显示模块的属性和数值

step 3 打开新的模块文件。在上面的操作界面中，打开一个新的模块文件"Sim2"，查看对应的菜单选项属性设置，如图8.58所示。

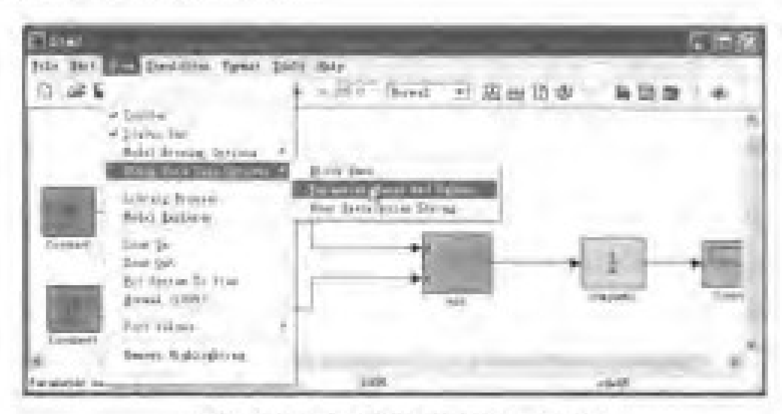


图8.58 查看新文件的显示属性



说明 从上面的实例中可以看出，对于不同的文件，该属性显示的选项在默认情况下是不会被选中的，对于不同的文件需要重新设置该属性。

除了上面的小技巧之外，还可以对模块进行各种操作，例如复制、移动、旋转等。对于这些操作，一般都可以使用多种方法来实现，本节只是提供一些比较常用或者比较便捷的操作方法，其他的方法请用户自行尝试。

8.3.3 复制和移动模块

这里描述的操作是指，用户从一个模块窗口中将某个模块复制或者移到另外一个模型窗口中。当用户在Simulink典型窗口中创建模块时，一般需要从模块库中选择所需的模块，然后将其拖动到窗口中。实质上，这种操作相当于从模型库中复制模块到相应的窗口中。因此，如果需要复制或者移动模块到窗口中，只需要简单的拖动操作就可以了。当然，也可以采用典型的复制对象方法，使用Copy和Paste菜单选项等，这里不重复介绍。



说明 复制后的模块对象和原来的模块具有完全相同的属性。如果在不同的窗口中复制模块，则对应的模块名称也相同；如果在同一个窗口中复制模块，这些模块使用名称后的数字加以区分。

旋转模块

在Simulink中，系统允许用户修改模块的方向属性，也就是在模型窗口中很便捷地实现模块的旋转。在默认情况下，信号会从左向右穿过模块，输入变量在模块的左侧，而输出变量则在模块的右侧。根据不同的系统要求，用户修改这种方向属性，主要的方法就是使用模型窗口中"Format"菜单下的"Flip Block"和"Rotate Block"子菜单选项：

- 选择"Flip Block"菜单命令可以将模块旋转180°；
- 选择"Rotate Block"命令可以将模块顺时针旋转90°。

图8.59显示上面两种命令的操作效果，在模块中的文字说明表明模块的方向属性。

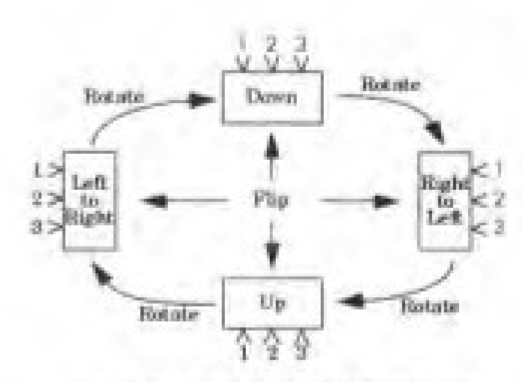


图 8.59 旋转模块的效果演示图



图 8.59 形象地显示了 Format 菜单选项下的 "Flip Block" 和 "Rotate Block" 子菜单选项的操作结果，其中 Flip 对应的箭头表示，使用 Flip 命令会产生对应的效果。

8.3.4 显示模块的属性数值

对于比较复杂的模块系统，如果能够直接在模块下方显示模块的属性数值，可以大大加强该系统的可读性和交流程度。为了达到上面的效果，可以选中对应的模块，然后打开对应的 Block Properties 对话框，在其中设置 "Block Annotation" 的属性数值。下面使用一个简单的实例来说明如何完成这种设置。

例 8.7 在模块的下方显示模块的属性数值。

step 1 打开前面保存的 "Sim5" 文件，然后选择该模型系统中的 "Signal Generator" 模块，单击鼠标右键，在弹出的快捷菜单中选择 "Block Properties" 菜单选项，打开该模块的属性对话框，如图 8.60 所示。

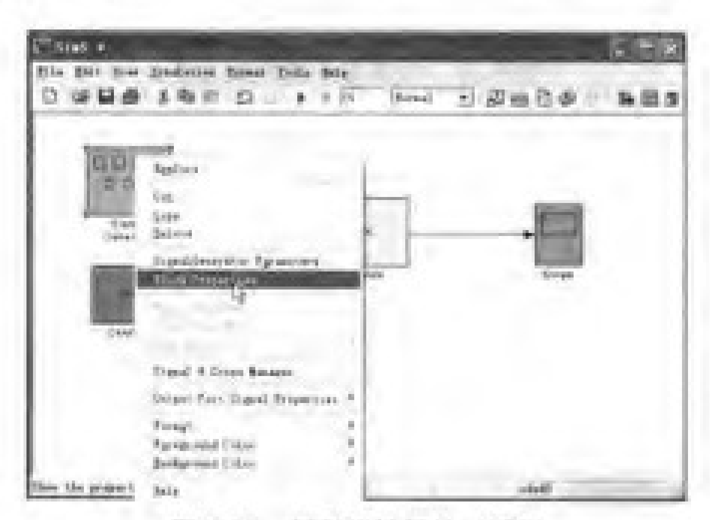


图 8.60 打开模块属性对话框

step 2 查看模块属性对话框。当选择该菜单选项后，系统显示默认的模块属性对话框，如图 8.61 所示。

step 3 设置显示属性的数值。选择对话框中的 "Block Annotation" 选项卡，然后在 "Enter text and tokens for annotation" 选框下面输入文字 "Amplitude=%<amplitude>, Frequency = %<frequency>"，单击 "Apply" 按钮，如图 8.62 所示。

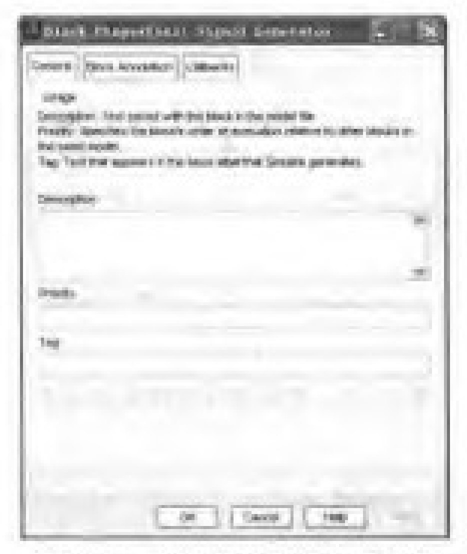


图 8.61　默认的模块属性对话框

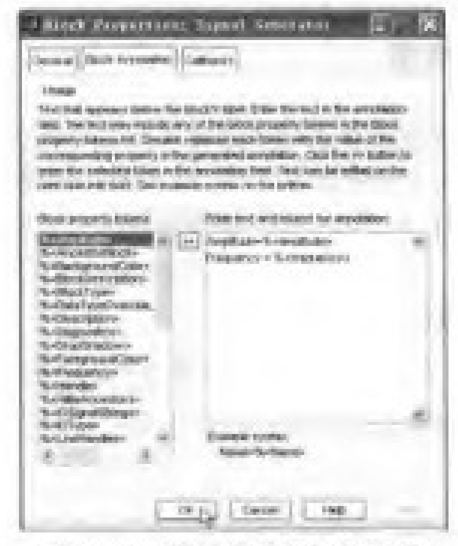


图 8.62　设置显示属性的选项

step 4 查看设置后的模块。将相应的属性选项设置完成后，单击 "OK" 按钮，关闭模块属性对话框，然后查看设置完成后的模块，如图 8.63 所示。

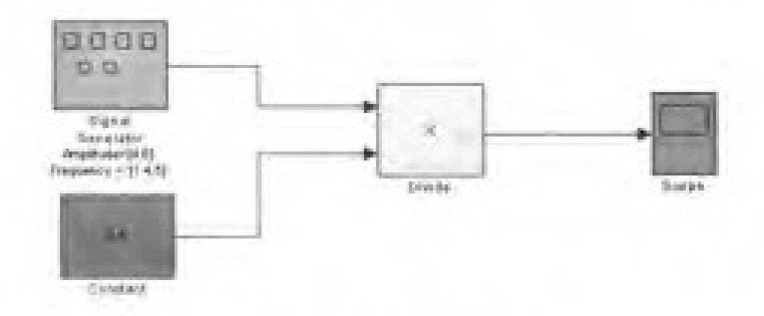


图 8.63　设置完成后的模块



从上面的步骤中可以看出，当设置了在模块下方显示的属性数值选项后，在模块的下方就会显示对应的属性名称和数值。关于设置选项的语法，限于篇幅就不详细展开了，感兴趣的读者可以查阅相应的帮助文件。

8.3.5 添加模块的阴影效果

在 Simulink 中，为了加强模块图标的外观特征，允许用户添加模块的阴影效果，使用的方法也比较简单。直接选中模块，然后选择菜单栏中的 "Format" ⇨ "Show Drop Shadow" 命令，就可以为选中的模块添加阴影效果，如图 8.64 所示。

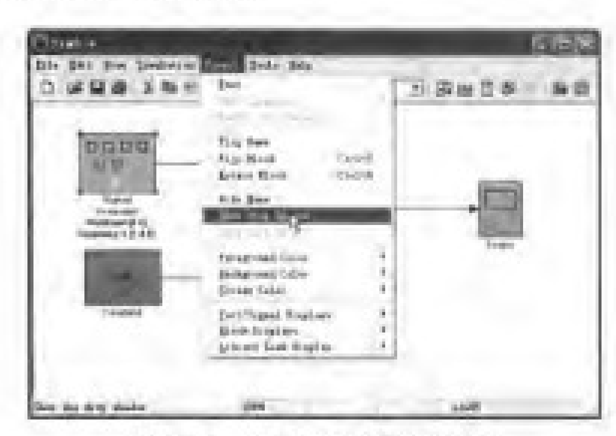


图 8.64　添加模块的阴影效果

当选择对应的菜单选项后，就可以添加模块的阴影效果，如图 8.65 所示。

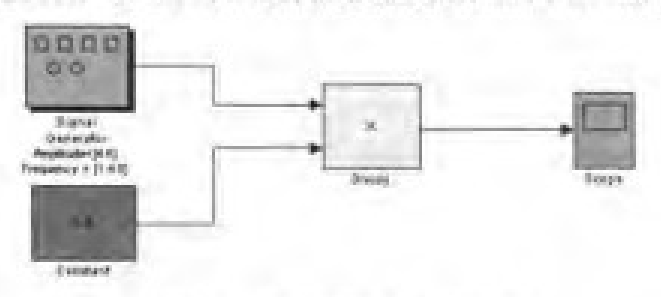


图 8.65　添加阴影效果后的模块



当用户为某个模块添加阴影效果后，对应的菜单选项也会发生变化，对应的 "Show Drop Shadow" 变成 "Hide Drop Shadow" 菜单选项。

8.3.6 操作模块名称

在 Simulink 中，所有的模块名称都是独一无二的，不能有重复，同时必须包含至少 1 个英文字母。对于模块名称的位置，MATLAB 设置一些默认属性，当模块的方向从左向右的时候，模块名称的默认位置在模块下方；当模块的方向是从上向下的时候，模块的名字在模块的左侧，如图 8.66 所示。



图 8.66　模块名称的默认位置



Simulink 会将向前斜线 "/" 认为是模块路径的分隔符。例如，路径 " vdp/Mu" 指定在 vdp 模型下的 Mu 模块。因此，在指定模块名称时，需要避免使用符号 "/"，否则 Simulink 会将对应的名称认为是路径。

在Simulink中，可以修改模块名称、编辑模块字体、改变模块名称的位置或者隐藏模块名称等各种操作，下面将详细介绍这些操作。

- **修改模块名称：** 单击模块名称，在名字的周围出现一个编辑框，可以在编辑框中输入新的模块名称；当修改完毕后，将鼠标移出编辑框，就可以完成修改。



如果修改模型库中某个模块的名称后，和该模块所有相关的连接也将会失去，因此在修改名称时需要特别留意这些问题。

- **设置模块名称的字体属性：** 选中模块，然后选择菜单栏中的"Format"⇒"Font"命令，打开"Set Font"对话框，在其中设置模块名称的字体属性，如图8.67所示。

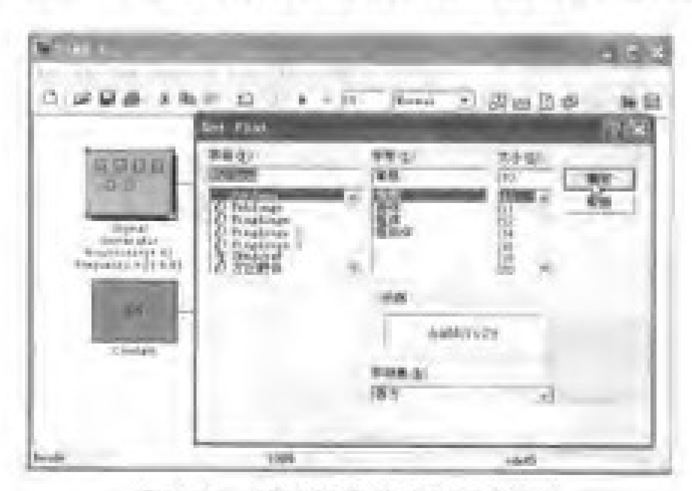


图8.67　设置模块名称的字体属性

- **修改模块名称的位置：** 单击模块名称，当模块名称周围出现编辑框后，直接使用鼠标拖动模块名称对象到合适的位置。
- **隐藏模块名称：** 选中模块名称，然后选择菜单栏中的"Format"⇒"Hide Name"命令，就可以隐藏对应模块的名称。同时，该菜单选项对应变为"Show Name"。



当旋转模块的时候，该模块对应的名称也会随之旋转，用户可以根据需要决定这些属性修改的次序。

8.3.7 显示模块的输出数值

在Simulink中，允许用户在运行模块的时候，在模块图标的上方显示仿真的模块输出结果的数值。设置的方法十分简单，下面以一个简单的案例来演示这种方法。

例8.8　显示模块在仿真过程中的结果数值。

step 1 打开前面章节中保存的"Sim5"文件，单击模型界面菜单栏中的按钮，设置显示模块运行的输出结果，如图8.68所示。

step 2 查看设置的结果。当完成上面的设置之后，将鼠标移动到"Signal Generator"模块上方，得到的结果如图8.69所示。

图 8.68 设置显示模块运行的输出结果

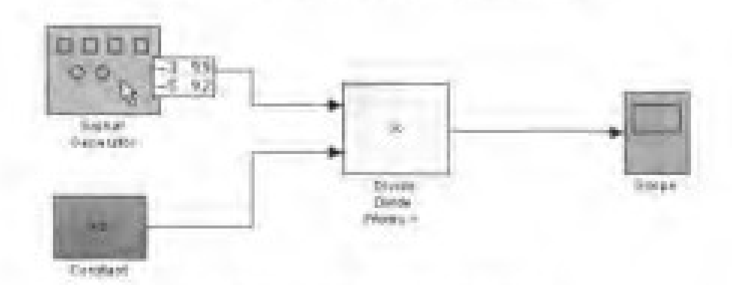


图 8.69 查看用户设置的结果

step 3 设置显示选项。选择菜单栏中的 "View" ⇨ "Port Values" ⇨ "Options" 命令，系统显示 "Sim5- Block Output Display Options" 对话框，可以在该对话框中设置关于显示选项的属性，如图 8.70 所示。

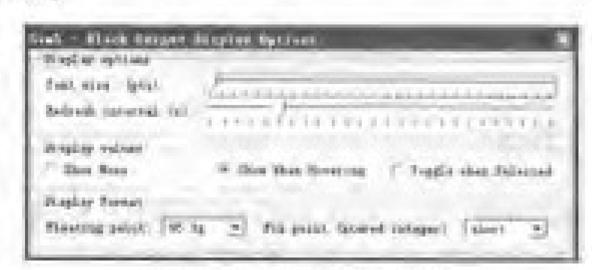


图 8.70 设置显示选项



如果希望加大显示的文字，可用鼠标将光标在 "Font size" 滚动条上向右移动；为了提高 Simulink 更新文字的速度，可以使用鼠标将光标在 "Refresh interval" 滚动条上向右移动。

8.3.8 连接线的分支

在 Simulink 中，连接线的主要功能是连接各种信号和模块，因此有时候连接线也会被称为信号线。在连接模块的时候，需要特别注意模块的输出、输入段和各模块之间的信号流向。关于连接线的常用操作包括设置连接线的方向、宽度、颜色、标识等主要内容，可以根据需要为 Simulink 系统模块中的连接线设置各种属性，下面详细介绍。

在某些复杂的仿真系统中，一个信号往往需要分送到不同模型的多个输入端口，此时就需要在该 Simulink 系统中绘制分支线（Branch line）。和连接线的其他绘制方法相比，分支线的绘制过程比较复杂，大致可以分为下面几个步骤：将光标指向分支线的起点，该起点就是某个信号线上的某个

点，按下鼠标右键，看到光标变成十字架形状后，拖动鼠标，直到分支线的终点处，松开鼠标，完成分支线的绘制。

为了更加直观地了解连接线的分支概念，选择MATLAB中自带的Demo文件 "ifsub.mdl"，在该文件中部分系统中出现了分支，如图8.71所示。

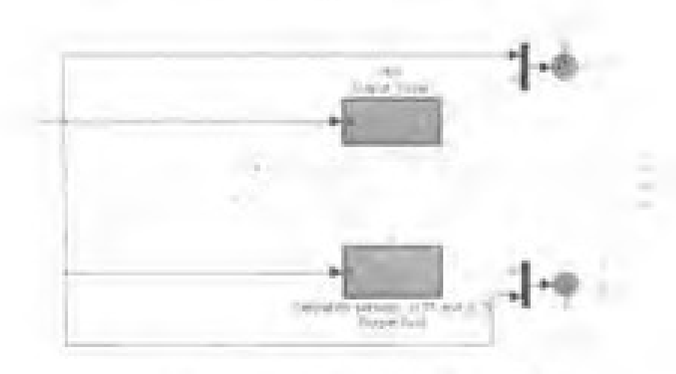


图 8.71　Simulink 中的分支结构

在上面的图中，高亮显示的就是一个分支结构，从 "Sine Wave" 信号源中输出的信号有四个流向，其中两个直接和上面两个 "Mux" 模块相连，其他两个则和程序模块相连，因此在上面创建的系统模块中，使用了连接线的分支结构。



在上面步骤中，为了更好地说明连接线的分支结构，选择包含分支节点的连接线，然后单击鼠标右键，在弹出的快捷菜单中选择 "Highlight to Destination" 选项，将这部分的连接线高亮显示。

8.3.9　移动连接线的节点

当使用Simulink创建各种系统时，有时需要将两个信号模块之间的连接线曲折，以让出空间，添加其他的模块。当折线的位置不能符合要求的时候，需要手动来修改节点的位置，修改的方法如图8.72所示。

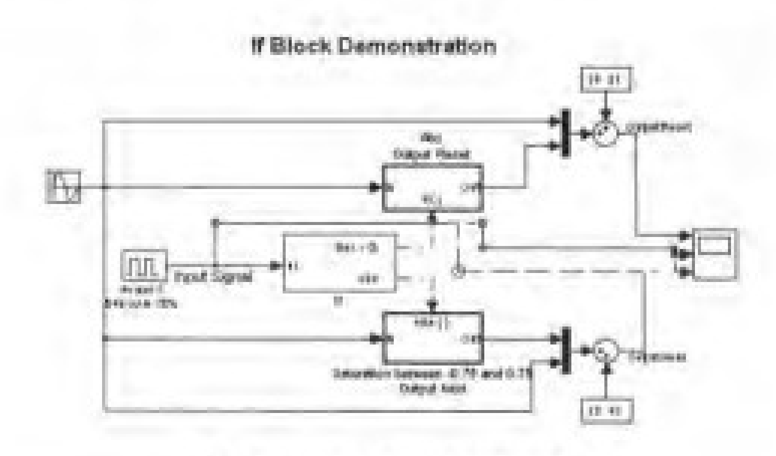


图 8.72　移动连接线的节点位置

如果希望修改节点位置，首先需要选中折线，然后将光标指向待移动的折点处，当光标变成一个小圆圈的时候，按下鼠标左键并拖动鼠标到合适位置。

8.3.10 彩色显示信号线

在 Simulink 所创建的离散系统中，允许用户采用多种采样频率。如果希望了解采用不同采样频率的模块以及信号线，可以为各个采样频率不同的信号线设置不同的颜色。这样，就可以十分直观地查看系统的频率。以 MATLAB 自带的 "enabsubs.mdl" 文件为例，说明如何使用彩色来显示信号线，如图 8.73 所示。

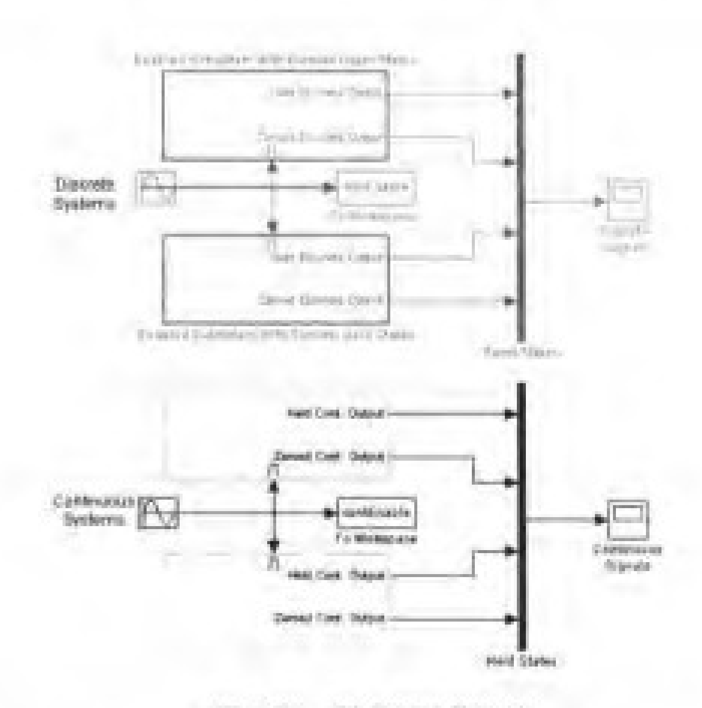


图 8.73　彩色显示信号线

在上面的例子中，选择菜单栏中的 "Format" ⇨ "Port/Signal Displays" ⇨ "Sample Time Color" 命令后，Simulink 自动根据用户设置的频率大小来使用不同的颜色，显示信号线和模块。

8.3.11 添加信号线标识

和在整个模型中添加注释的原理类似，对于信号线，为了让其他用户更加容易理解信号线的信息，可以为该信号线添加标识。添加信号线标识的方法并不复杂，只需双击对应信号线，系统会弹出一个空白的文字填写框，在其中输入文本，作为该信号线的标识。为了加强对标识的直观了解，以 MATLAB 自带的 "busdemo.mdl" 文件为例，用户可以查看该模型中的信号线标识，如图 8.74 所示。

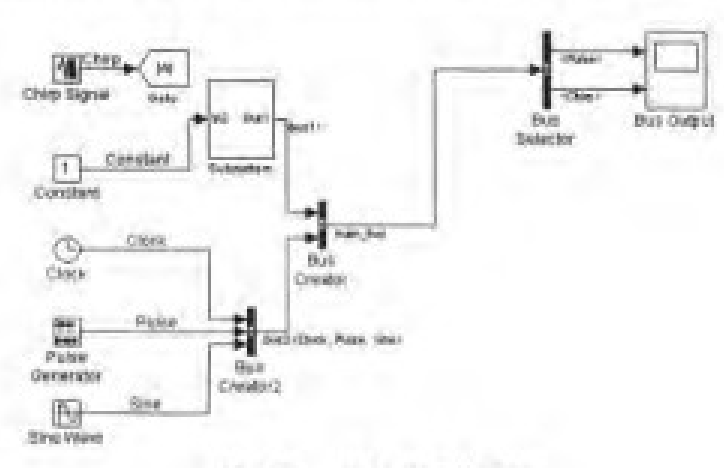


图 8.74　添加信号线标识



对于信号线标识，同样可以进行复制、修改、移动或者删除等，这些操作都十分简单，这里就不重复介绍了，请读者自行查看对应的帮助文件。

8.3.12 设置连接线的属性

作为Simulink中的一个重要组件，可以像其他组件一样，设置连接线的属性。选中某个连接线，然后单击鼠标右键，在弹出的快捷菜单中选择"Signal Parameters"选项，就可以打开该条连接线的属性对话框。

还是以MATLAB自带的"busdemo.mdl"文件为例，选择该模型中的某条连接线，打开对应的属性对话框，如图8.75所示。

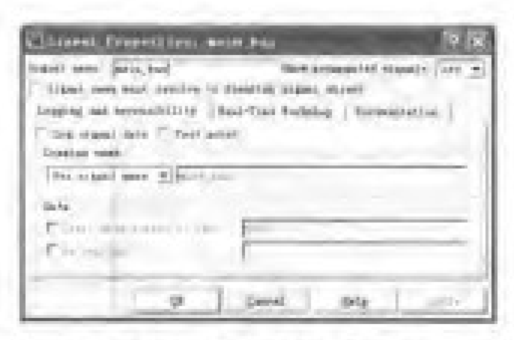


图8.75 设置连接线的属性



关于连接线的具体属性内容，这里就不详细展开介绍了，感兴趣的读者可以自行查阅相应的帮助文件。

8.4 Simulink的信号

在Simulink中，信号被定义为动态系统的输出，这些动态系统由Simulink图形化系统中的模块来表示。Simulink结构图中的连接线代表由模块图标定义的信号之间的数学关系，例如，连接模块A的输出端和模块B输入端的连接线代表模块B的输出量取决于模块A的输出量。也就是说，Simulink中的信号间的关系是建立在数学上的，代表数学逻辑关系，而不是图标所显示的外观物理上的联系。



笔者曾经阅读过关于MATLAB的Simulink的一个比喻，将Simulink中的信号比喻为沿着电话线传输的电子信号，这种比喻似乎比较形象，但是笔者并不认同这种比喻，因为这种比喻会误导读者，认为Simulink中的信号之间的关系是物理关系，而不是逻辑关系。

8.4.1 创建信号

在Simulink中，信号是一个完整而且复杂的系统内容。在详细介绍各种类型的信号之前，本小节将首先介绍关于信号的基础知识，读者可以对信号有个直观的了解。

可以直接使用模块库中的模块，创建自己的信号。例如，需要创建一个随着时间正弦变化的某种信号，直接从Simulink模块库中选择Sine模块添加到用户创建的模型中。除了使用这种方法之外，还可以使用"Signal & Scope Manager"对话框来创建自己的信号系统，而不使用Simulink提供的模块库中的信号。



上面文章中提到的"Signal & Scope Manager"对话框，在用户使用信号、创建信号中会经常使用到，将在后面详细介绍。

8.4.2 添加信号标签

和连接线类似，可以为信号添加标签来代表信号内容。如果信号是"Virtual Signal"，而且其对应的"Show propagated signals"属性被选中，则该信号的标签显示的是组成该信号所有信号的名称。可以打开某个信号的"Signal Properties"对话框，然后在其中为信号添加标签。

当某个信号包含标签后，用户可以直接选中该标签，输入新的标签名称，然后按"Enter"键，完成名称的修改。

下面以MATLAB自带的"busdemo.mdl"文件为例，为添加在模型中的"Chirp Signal"信号添加名称，如图8.76所示。

图8.76 添加信号标签

8.4.3 显示信号数值

当创建某个信号对象后，可以使用信号接收模块或者"Signal & Scope Manager"对话框显示仿真过程中信号的数值。例如，使用"Scope"模块或者"Signal & Scope Manager"绘制随时间变化的信号曲线。读者可以查阅"Block Libraries"的相关帮助文件来查看关于信号接收模块（Sinks）的信息。



上面提到的"Signal & Scope Manager"对话框，在用户使用信号，创建信号中会经常使用到，将在后面详细介绍。

8.4.4 复数信号

默认情况下在Simulink中使用的信号都是实数，但在实际情况中经常需要处理复数信号。在

Simulink 中通常使用下面的方法来处理复数信号的模型：

- 向模型中添加一个 Constant（常数）模块，将其参数设置为复数；
- 分别生成复数的虚部和实部，然后使用 Real-Imag to Complex 模块将这些部分复合为一个复数对象；
- 分别生成复数的幅值和幅角，使用 Magnitude-Angle to Complex 模块将这些部分复合为一个复数对象。

例 8.9 在 Simulink 中处理复数信号。

step 1 打开空白模型窗口界面，向模块窗口中添加如图 8.77 所示的模块，使用不同的方法来创建复数。

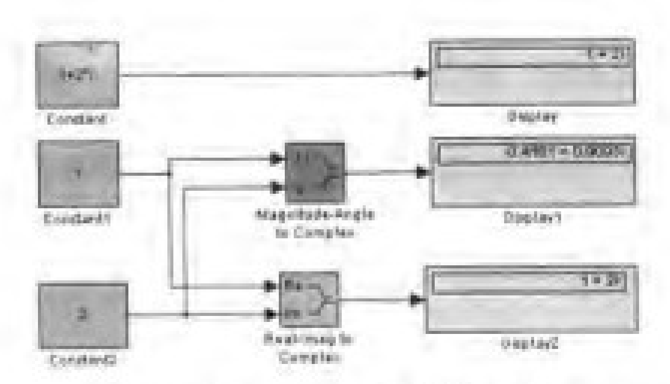


图 8.77 使用不同的方法来创建复数信号



在复数处理模块中，分别显示复数的各个参数，可以使用这种参数来分别处理参数。在 Magnitude-Angle to Complex 模块中，|| 表示的是复数的模，∠则表示复数的幅角，这两个参数不能混淆。

step 2 将复数信号再次转换为实数信号。在上面的模型窗口中依次添加 Complex to Real-Imag 和 Complex to Magnitude-Angle 模块，将上面步骤中合成的复数信号转换为实数信号，如图 8.78 所示。

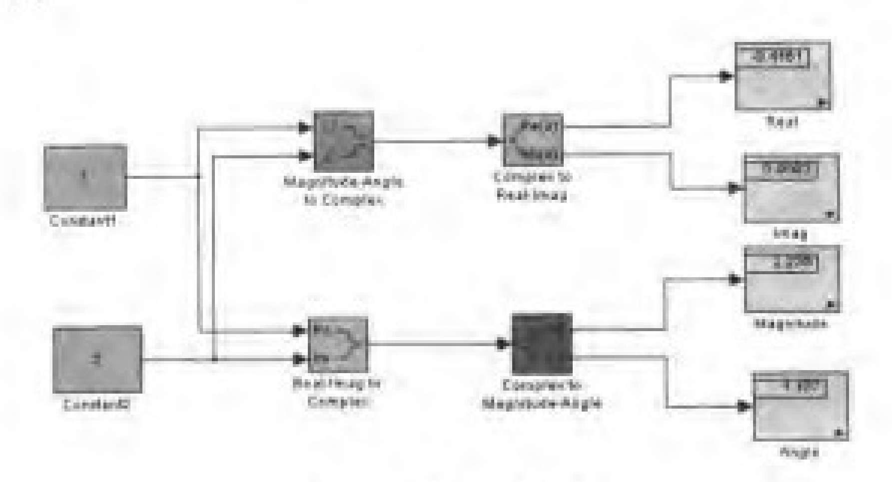


图 8.78 转换为实数信号

8.4.5 虚拟信号

虚拟信号是使用图形来代表其他信号的一个信号。在Simulink中的某些模块，例如Bus Creator、Inport和Outport blocks等都可以产生虚拟信号。虚拟信号完全是图形化的实体，没有数学或者逻辑上的作用，当Simulink 在进行系统仿真的时候会直接忽略它们。当运行或者更新模型时，系统使用信号传播来直接确定该虚拟信号代表的信号实体，保证整个系统的正常运行。上面的介绍文字比较抽象，下面使用一个简单的实例来说明虚拟信号的概念。

例 8.10 使用一个简单的案例来演示虚拟信号的概念。

step 1 打开空白模型窗口界面，向窗口界面中添加“常数”模块，添加Bus Creator、Bus Selector 等虚拟信号模块和增益模块，最后添加“Display”模块，如图 8.79 所示。

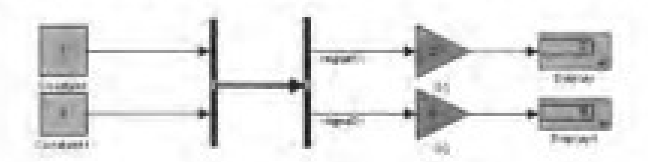


图 8.79 添加虚拟信号模块

在图 8.79 模块中，添加了两个虚拟信号模块，得到的结果和没有添加虚拟模块一样，分别将两个参数增益 2 倍或者 3 倍，得出最后的增益数值。也就是说，图 8.79 模块系统和图 8.80 所示的模块系统效果完全相同。

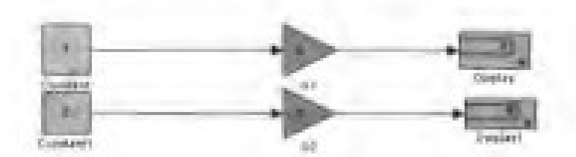


图 8.80 清除虚拟信号后的系统

step 2 修改信号名称。双击上面系统中的“Bus Creator”模块，打开对应的对话框，在其中修改信号的名称，如图 8.81 所示。



图 8.81 修改信号的名称



在默认情况下，"Bus Creator" 模块输出信号的名称为signal 1、signal 2，为了让整个系统更加简洁，将上面的名称改为S1、S2。

step 3 修改信号输出的次序。双击上面系统中的 "Bus Selector" 模块，打开对应的对话框，在其中修改信号输出的次序，如图 8.82 所示。



图 8.82　修改信号输出的次序

step 4 重新运行整个系统。完成上面的修改后，单击对话框中的 "OK" 按钮，关闭对话框，然后重新运行该仿真系统，得到的结果如图 8.83 所示。

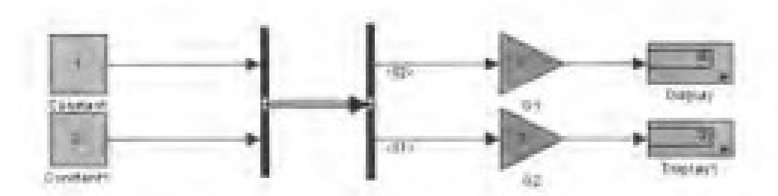


图 8.83　重新运行系统

从上面的步骤中可以看出，当将 "Bus Selector" 虚拟模块的输出信号向量修改以后，结果就发生了变化，得到的结果是常数 1 增益 3 得出 3，常数 2 增益 2 得出 4。



在Simulink 中，虚拟信号模块可以代表虚拟信号或者非虚拟信号。例如，可以使用 "Bus Creator" 模块来连接多个虚拟信号和非虚拟信号。

8.4.6　控制信号

在 Simulink 中，控制信号是指由控制其他模块执行的某个模块发出的信号。例如，某个函数调用或者执行子系统。关于该信号的详细内容将会在后面的章节中介绍，在这里以一个比较简单的实例来说明控制信号的使用方法。

例 8.11　使用 While 控制子系统，从 1 到自然数 N 的叠加和不超过 100 的最大自然数。

step 1 打开空白模型窗口界面，向界面中添加各模块，得到的结果如图 8.84 所示。

图 8.84 添加 While 子系统

其中，上面 "Constant" 模块表示的是叠加的最大数值 100，下面 "Constant" 模块表示的是叠加的初始数值 0。

step 2 设置关系表达式模块的属性。"Relational Operation" 模块表示的是最后关系中叠加数值不超过 100，因此应该选择 "<"。完成上面的设置，需要设置 "Relational Operation" 模块对应的参数属性，如图 8.85 所示。

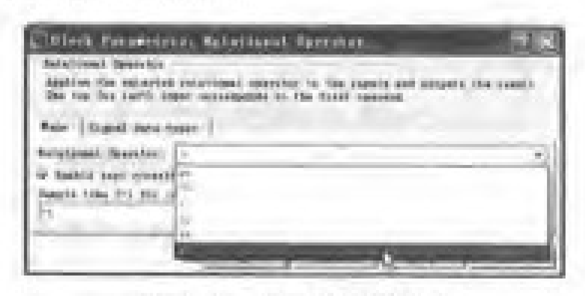


图 8.85 设置关系表达式

While Subsystem 是本系统的核心模块，需要在后面的步骤中为其添加对应的子系统。在系统的最右侧，添加了 "Display" 模块，显示该系统计算的结果。

step 3 双击 "While Subsystem" 模块，打开该子系统的编辑界面，然后在其中添加该子系统的各模块，如图 8.86 所示。

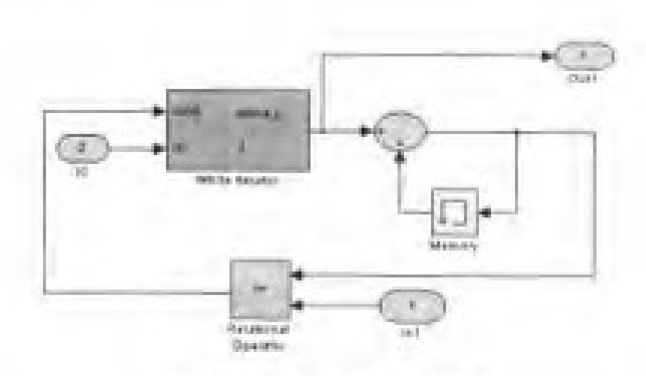


图 8.86 编写子系统的模块

step 4 设置 "While Iterator" 模块的属性。设置 "While Iterator" 模块的属性的目的是有效地控制整个循环，打开模块属性对话框，在其中设置参数值，如图 8.87 所示。

在上面的参数对话框中，"Maximum number of iteration" 文本框中输入的是 "－1"，表示允许程序持续循环，直到关系条件为假。

step 5 设置 "Memory" 模块的属性。双击 "Memory" 模块，打开对应的模块属性对话框，在其中设置属性，如图 8.88 所示。

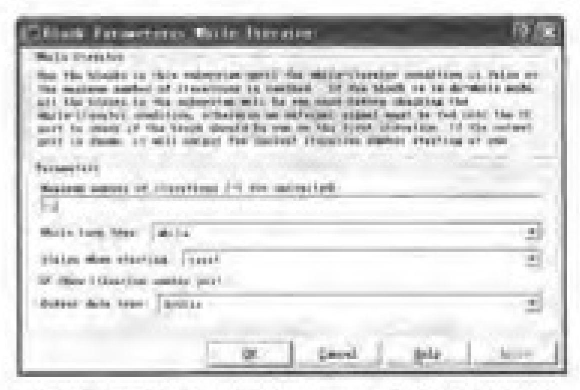


图 8.87 设置“While Iterator”模块的属性



图 8.88 设置“Memory”模块的属性

step 6 运行整个仿真系统，得出仿真计算的结果。完成上面的所有参数属性设置后，单击模型界面中的“开始仿真”按钮，运行整个系统，得到的结果如图 8.89 所示。

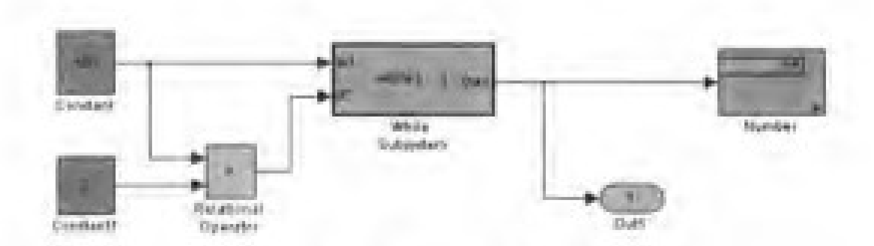


图 8.89 得出仿真结果

从上面的结果可以看出，使用上面程序模块得到的结果为 13，也就是说，13 是从 1 到 N 的数值叠加后总和小于 100 的最大自然数。这个结果可以使用简单的数学知识得到验证。

根据数学知识，$1+2+3+\cdots+n=\frac{n(n+1)}{2}$，根据上面的数学公式，当 $n=13$ 的时候，其叠加求和为 $1+2+3+\cdots+13=\frac{13\times(13+1)}{2}=91$。所以就验证了上面的程序代码结果是正确的。

step 7 修改程序的约束条件，重新运行仿真系统。将叠加求和的上限改为 300，然后重新运行仿真系统，得到的结果如图 8.90 所示。

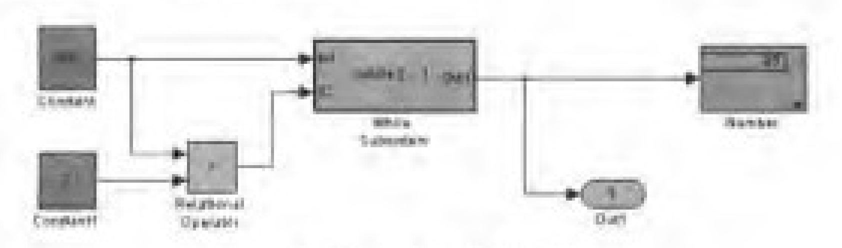


图 8.90 修改程序条件

熟悉编程的读者也许会想到，上面仿真系统的效用和下面的程序代码完全相同：

```
max_sum = 100;
sum = 0;
iteration_number = 0;
cond = (max_sum > 0);
while (cond != 0) {
    iteration_number = iteration_number + 1;
    sum = sum + iteration_number;
    if (sum > max_sum OR iteration_number > max_iterations)
        cond = 0;
}
```

从上面的演示过程可以看出，控制信号实质上在功能上相当于程序代码，用好控制信号可以很方便地演示各种程序功能。



上面的Simulink实例对于初学Simulink的用户而言，比较复杂，关于上面实例的信息将在后面章节对应的位置详细介绍。

8.4.7 信号总线（Signal Buses）

在Simulink中，信号总线是指一个组合信号，该信号由一束连接线所代表。在外观上，其相当于由包装带所捆绑的一束电线。信号总线的组件可以包含不同的数据类型，同时其组件也可以是复合信号。

在Simulink中，可以使用Bus Creator模块（或者Inport模块）来创建信号总线，使用Bus Selector模块来接收信号总线，在Simulink的模块库浏览器中，上面两个模块都被列在"Commonly Used Blocks"模块库之下。在前面介绍"虚拟信号"的小节中，曾经接触过"信号总线"对象，如图8.91所示。

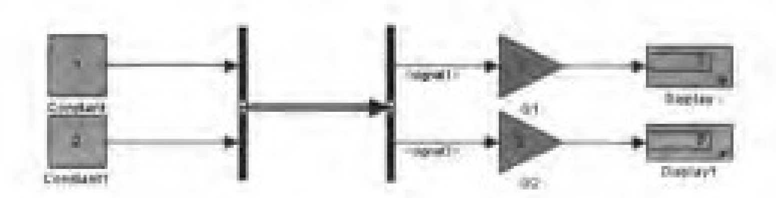


图 8.91　信号总线对象

由于信号总线的输入信号比较复杂，如果希望直观地了解到某个信号总线的信号个数，可以选中该总线对象，然后选择"Format"⇨"Port/ Signal Displays"⇨"Signal Dimensions"命令，就可以查看信号总线的组成信号个数，如图8.92所示。

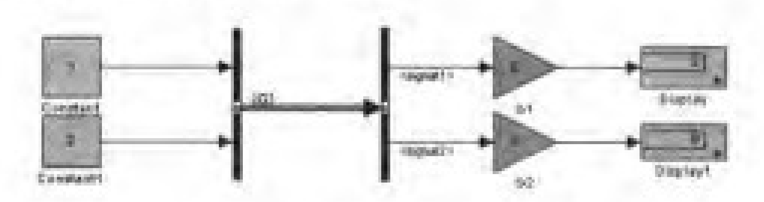


图 8.92　显示信号总线的信号个数

在详细介绍信号总线之前，有必要提醒读者：信号总线上传输的可能是虚拟信号也可能不是虚拟信号。在系统进行仿真的过程中，和虚拟信号总线相连的模块将会从分配给其组件信号的内存空间中读取输入变量，这些组件信号一般会保存在不连续的内存空间中。与此不同，和非虚拟信号总线相连的模块则从由 Simulink 保存的组件信号副本中取出输入变量的数值，这些信号的信息被保存在 MATLAB 分配给信号总线的连续内存空间中。



在 Simulink 中，某些属性，例如模型引用，需要使用非虚拟信号模块，其他则需要使用虚拟信号模块。

在 Simulink 中，使用 Bus Creator 和 Inport 模块创建的信号总线对象在默认情况下是虚拟信号；如果创建非虚拟信号，则需要在其对应的参数对话框中将其输出的属性设置为非虚拟的。



除了上面文字中介绍的方法，还可以使用 Signal Conversion 模块在虚拟信号和非虚拟信号之间进行转换。

在 Simulink 中，不是所有的信号模块都可以和信号总线相连的，能够和信号总线相连的模块是那些同时能够通过虚拟信号、非虚拟信号的模块。所有的虚拟信号都可以和信号总线连接，同时，下面的所有非虚拟信号模块也可以和信号总线相连：

- Memory
- Merge
- Switch
- Multiport Switch
- Rate Transition
- Unit Delay
- Zero-Order Hold



限于篇幅，关于上面这些信号模块的详细知识和内容这里就不展开说明了，感兴趣的用户可以自行查阅相应的帮助文件。

在一般情况下，Bus Creator 模块是虚拟模块，因此该模块会接收信号总线作为输入变量；如果该模块处于条件执行状态或者子系统状态下，或者其组件和某个子系统的输出端口直接相连时，该模块是非虚拟模块。在此情况下，只有其组件有相同的数据类型时，该模块才能接收信号总线对象的数据；如果组件包含不同的数据类型，当进行对应的仿真时，Simulink 会中止仿真过程，并显示错误信息。为了解决上面的问题，需要添加 "Signal Conversion" 模块，进行信号数据类型的转换，下面举例说明。

例 8.12 演示信号总线的数据类型传递问题。

step 1 打开空白模型界面，在其中添加系统的模块，如图 8.93 所示。

step 2 修改模块的数据类型。在图 8.93 系统模块中，首先添加两个 "Constant" 模块，为了演示信号总线的数据传递的问题，将其中一个模块的输出变量的数据类型改为 "int16"，如图 8.94 所示。

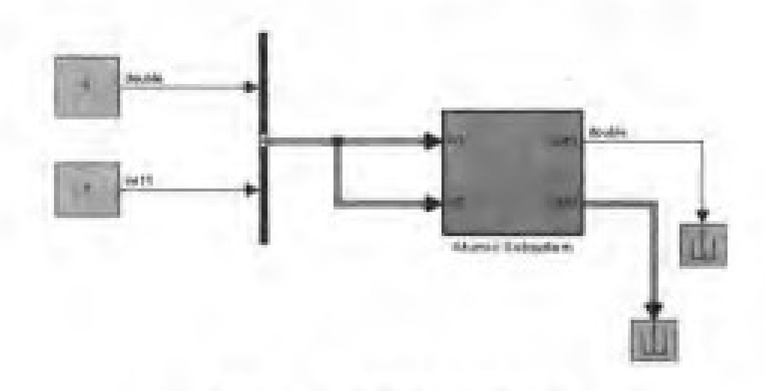


图 8.93　添加程序系统的模块

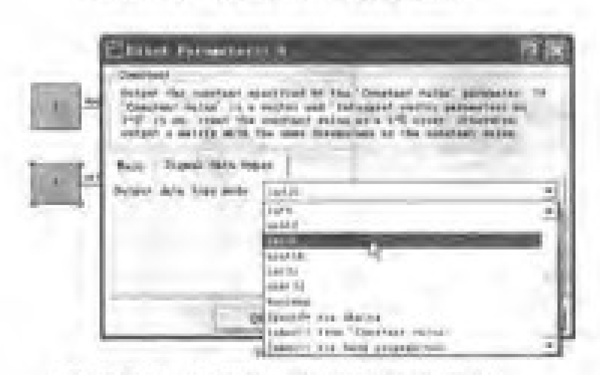


图 8.94　修改模块输出变量的属性

step 3 建立子系统模块。双击图 8.93 系统中的 "Atomic Subsystem" 模块，打开新的模块编辑窗口，在其中建立对应的子系统模块，如图 8.95 所示。

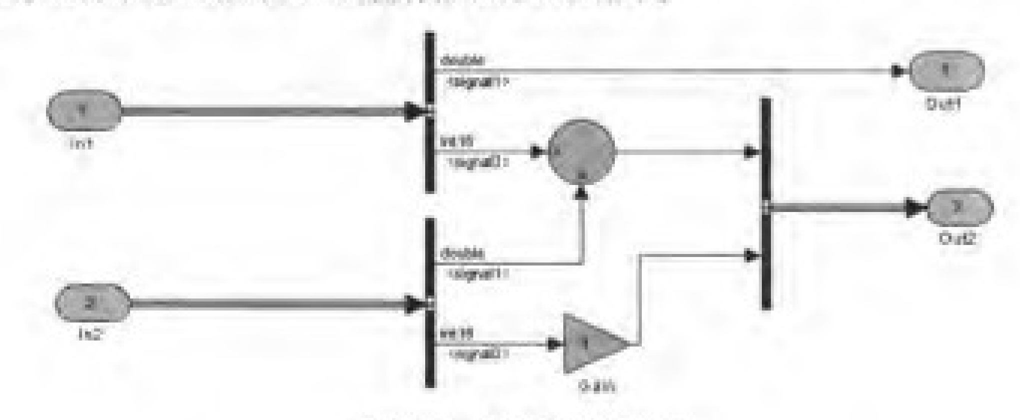


图 8.95　创建子系统模块

在上面的模块中，将系统的两种输入变量分别进行对应的数学计算，得到新的数值。之所以设计这样的模块结构，是为了在后面的步骤中演示如何处理数据类型的冲突。

step 4 保存模型，运行仿真。保存上面步骤中设计的子系统，然后运行系统的仿真，得到的系统提示如图 8.96 所示。

根据上面的系统提示信息，系统中止系统仿真的主要原因在于输入变量的数据类型不能协调，而 Bus Creator 模块不能接收该混合数据类型，因此系统不能进行数据传递，直接中止仿真过程。

step 5 单击对话框的 "Close" 按钮，返回到子系统模块界面中，向其中添加信号转换模块，如图 8.97 所示。

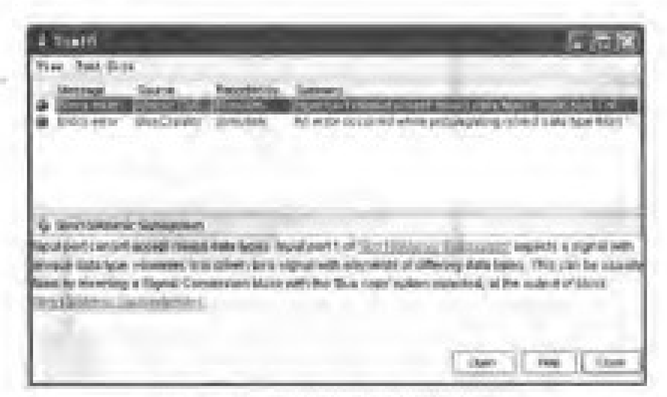


图 8.96　系统提示的错误信息

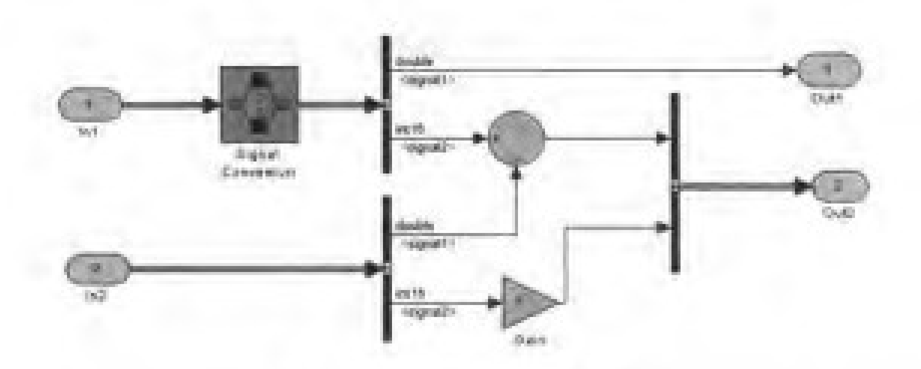


图 8.97　添加信号转换模块

step 6 设置 "Signal Conversion" 模块的属性。双击该模块，打开对应的属性对话框，将输出变量的数据类型选择为 "Bus copy"，如图 8.98 所示。

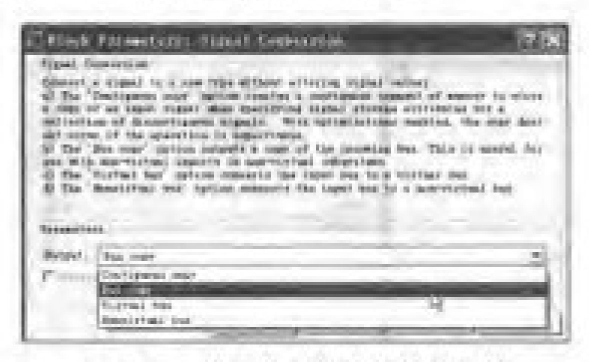


图 8.98　设置信号转换模块的属性

step 7 设置 "Gain" 模块的 "Fixed-Point" 属性。选择上面的 "Gain" 模块，然后选择 "Tools" ⇨ "Fixed-Point Settings" 命令，打开 "Fixed-Point Settings" 对话框，设置对应的属性选项，如图 8.99 所示。

在上面的对话框中，将模块的 "Logging Mode" 属性设置为 "Force Off"，将模块的 "Data type override" 属性设置为 "True doubles"。完成上面的设置之后，可以重新运行该仿真系统，发现 Simulink 可以正常运行该系统。



说明　上面的实例并没有得出实质的程序结果，只是为了演示信号总线之间的数据传递问题。关于上面的属性设置问题，将分别在后面的章节中详细介绍。

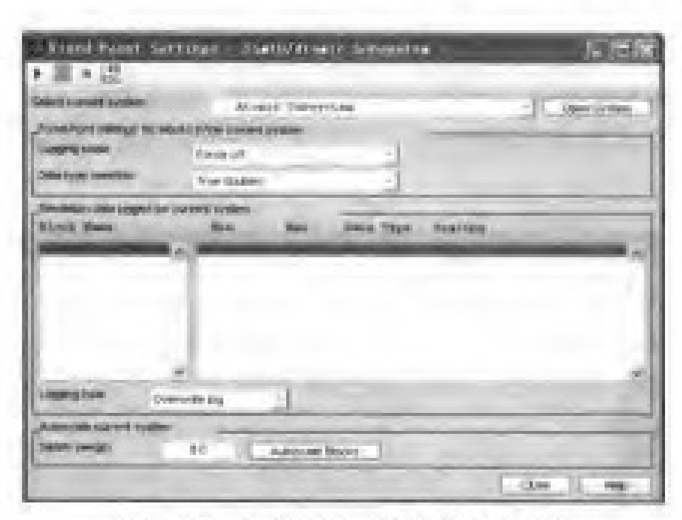


图 8.99 设置 "Fixed-Point" 属性

8.4.8 信号组

Simulink 提供可以相互交换的信号组工具，只需要从 "Sources" 模块库中的 "Signal Builder" 模块，将其添加到新的模块编辑器中，就可以创建最简单的，也就是默认的信号组对象，创建的系统模块如图 8.100 所示。

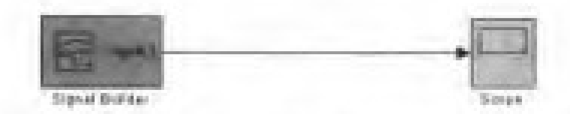


图 8.100 创建的系统模块

双击 "Scope" 模块，就可以查看系统默认的信号组，如图 8.101 所示。

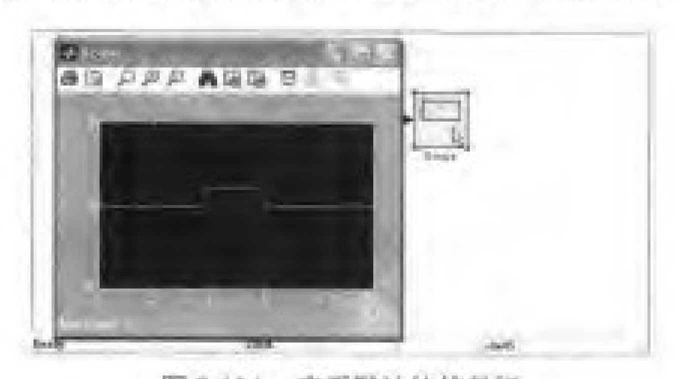


图 8.101 查看默认的信号组



从图 8.101 中可以看出，在默认情况下 Simulink 提供的信号是一个能够产生方形脉冲波的单个信号。

用户可以使用系统提供的 "Signal Builder" 对话框完成信号的编辑工作，包括添加新的信号、修改原有信号的属性、选择产出的信号对象等。只需双击上面系统中的 "Signal Builder" 模块，就可以打开对应的属性编辑器，如图 8.102 所示。

在该对话框中，可以创建和编辑由 "Signal Builder" 模块创建的信号组，其中包含下面几个重要的控件。

✦ **"Group" 面板**：在该面板中显示该模块所代表的可交换的信号源，为每组信号显示用于编辑的波形属性。该面板的选项卡中显示信号组的名称，在该面板中每次只能看到一组信号。可以选择包含信号组名称的选项卡，显示对应的信号组。同时，模块将会输出在该面板中可见的信号信息。

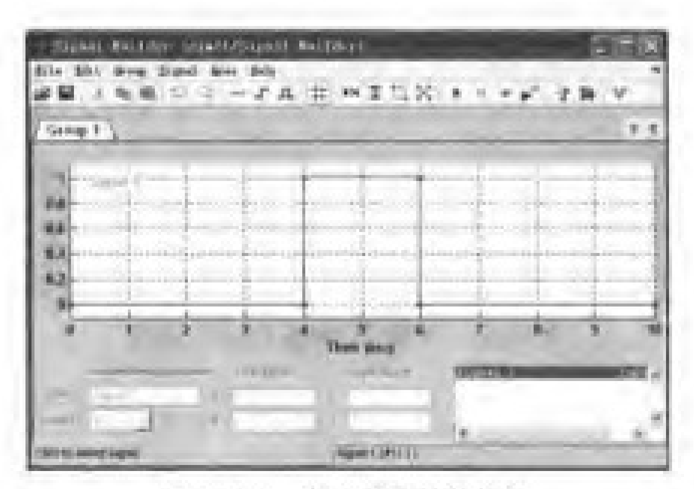


图 8.102　信号组属性编辑器

✦ **信号坐标轴**：在该信号组中，信号分别出现在独立的坐标轴中，这些坐标轴拥有相同的时间轴。之所以设计成这样的坐标系统，是为了方便用户比较不同信号随时间变化的情况。在 "Signal Builder" 对话框中，系统为每组信号产生自适应的坐标系统，显示最佳的波形。可以使用 "Signal Builder" 对话框中的 "坐标轴" 菜单，修改所选坐标轴的时间范围（T）和振幅（Y）的数值。

✦ **信号列表**：该控件中显示当前所选信号组的信号名称和可见性。选择该列表中的某个条目就可以选中对应的信号，双击该条目名称就可以在 "Group" 面板显示（或者隐藏）该信号的波形。

在 "Signal Builder" 对话框中，可以编辑信号组和编辑信号的各种属性。完成这些编辑工作的操作都十分简单，这里就不详细介绍了。下面主要介绍如何在该对话框中直接编辑信号的波形，为了简化说明，还是以默认的信号组为例。

例 8.13　显示如何编辑信号组对象。

step 1 打开系统默认的信号组对象的编辑对话框，使用鼠标在信号波形的任意位置单击，选中该波形对象，此时对话框中会显示波形的各个数据点，如图 8.103 所示。

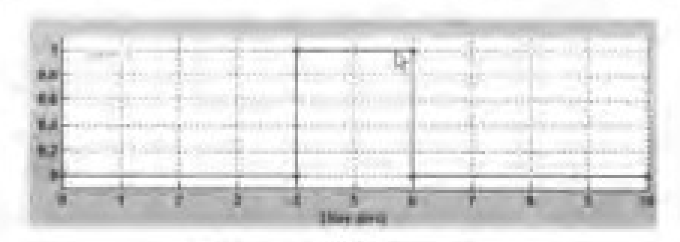


图 8.103　选中波形对象



在 "Signal Builder" 对话框中，许多菜单选项只有在选中波形对象后，才会被激活，因此在介绍如何编辑信号组对象之前，必须首先选中该波形对象。

step 2 设置Y坐标轴（振幅）网格线的属性。选中波形对象后，选择菜单"Axes"⇨"Set Y snap grid"⇨"0.2"命令，将波形的Y轴单位网格数值设置为0.2，如图8.104所示。

在"Signal Builder"对话框中，每个波形坐标轴中都包含一个无形的网格线，以方便用户确定波形数据点的准确位置坐标。当系统生成一个波形坐标轴的同时也会产生该网格线。当用户需要移动某个波形数据点或者波形片段位置的时候，Signal Builder将数据点或者片段移动到网格线上的数据点上。

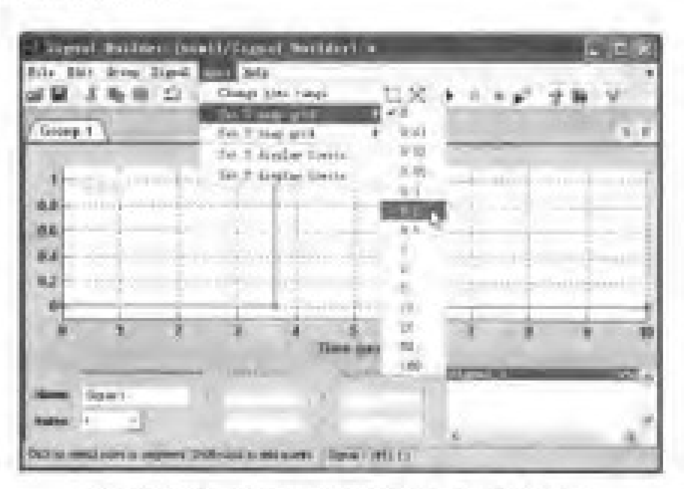


图8.104　设置振幅坐标轴的网格属性



在设置网格线的最小单位数值时，需要注意的问题是，当将网格线的最小单位设置得最精细时，就越容易移动波形片段或者数据点，但是就越难确定波形数据点的准确坐标数值。因此，在设置最小单位网格线时需要谨慎。

step 3 设置T坐标轴（时间）网格线的属性。选中波形对象后，选择菜单"Axes"⇨"Set T snap grid"⇨"0.1"命令，将波形的T轴单位网格数值设置为0.1，如图8.105所示。

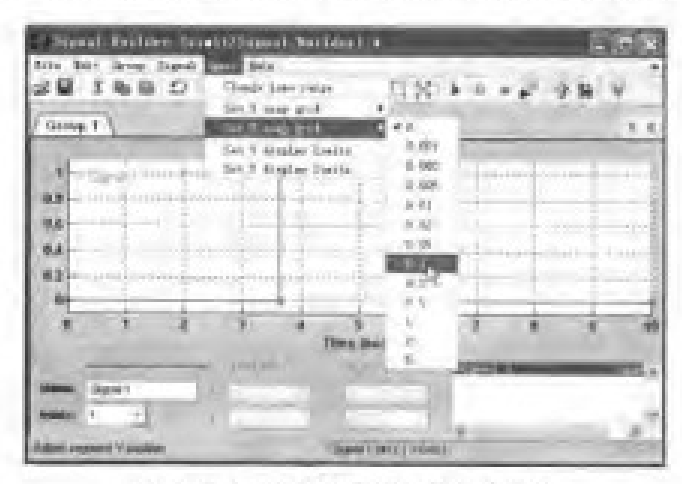


图8.105　设置时间轴的网格属性



从上面的设置过程中可以看出，在默认情况下，Y轴和T轴的网格线最小单位数值都是0。这样设置的目的在于方便用户修改波形的位置。

step 4 上下移动波形片段。选中波形某个图形片段，使用鼠标将波形图形片段上下移动，如图8.106所示。从波形片断图形可以看出，由于在前面步骤中将波形的Y轴单位网格数值设置为0.2，因此在上下移动波形图形片段时，最小的移动单位就是0.2。如果采用的是默认数值0，就可以自由移动波形。

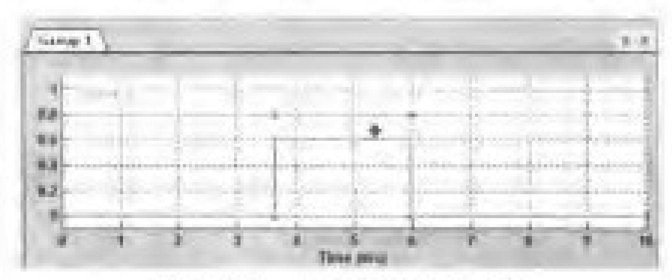


图 8.106　上下移动波形片段

step 5 左右移动波形片段。选中波形的某个图形片段，使用鼠标将波形图形片段左右移动，如图8.107所示。

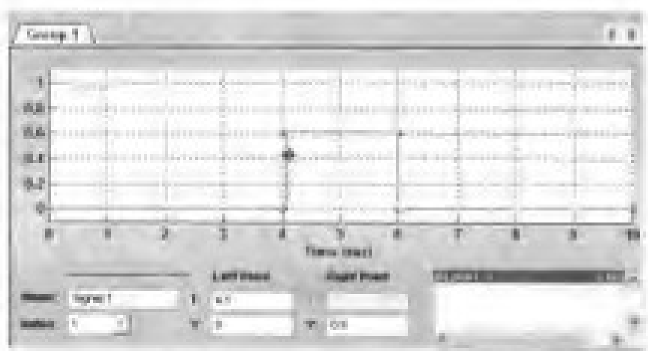


图 8.107　左右移动波形片段

从上面的步骤中可以看出，由于在前面步骤中将波形的T轴单位网格数值设置为0.1，因此在左右移动波形图形片段时，最小的移动单位就是0.1。从对话框的坐标数值中可以看出，T的数值被改为4.1，这是最小的移动单位。

step 6 修改波形数据点的坐标数值。选中上面波形对象中某个数据点，然后在下面的坐标轴数值框中输入该数据点的坐标数值，如图8.108所示。

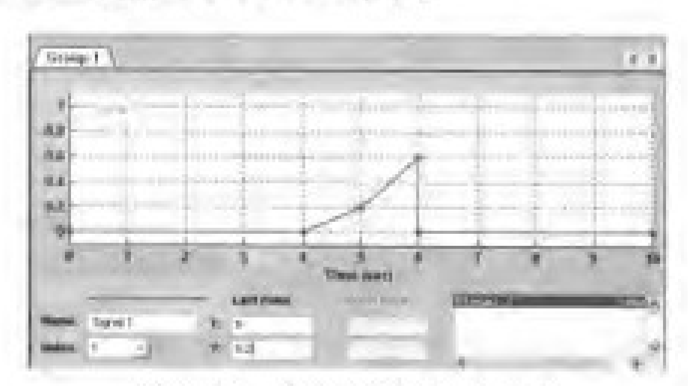


图 8.108　修改数据点的坐标数值



在“Signal Builder”对话框中，如果希望选中某数据点，可将鼠标移到该数据点上方单击鼠标左键，直到该数据点上出现圆圈为止。

step 7 修改波形的线型。选中波形图形，选择菜单栏中的 "Signal" ⇨ "Line style" ⇨ "Dashed-dotted" 命令，修改该图形的线型为点虚线类型，如图 8.109 所示。

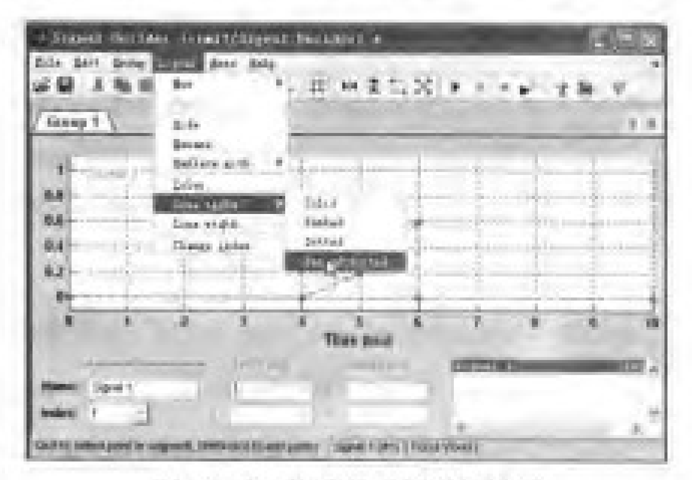


图 8.109 修改波形图形的线型



除了修改波形的线型之外，还可以修改波形的线宽，只需要在选中波形对象中，选择 "Signal" ⇨ "Line width" 中的相应命令，设置新的线宽。

step 8 修改波形的时间范围。选中波形图形，选择菜单栏中的 "Axes" ⇨ "Change time range" 命令，打开 "Set the total time range" 对话框，在其中设置波形的时间范围，如图 8.110 所示。

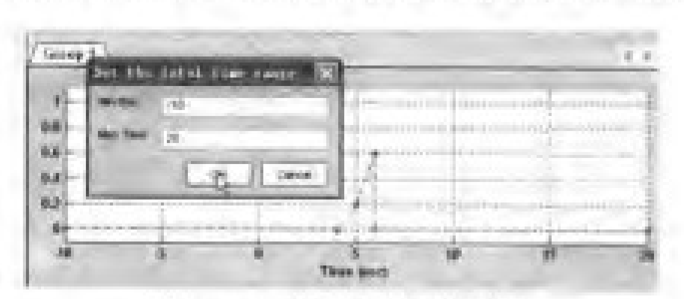


图 8.110 设置波形的时间范围

step 9 将波形输出到工作空间中。选择菜单栏的 "File" ⇨ "Export to workspace" 命令，打开 "Export to workspace" 对话框，将波形输出到工作空间的变量中，如图 8.111 所示。

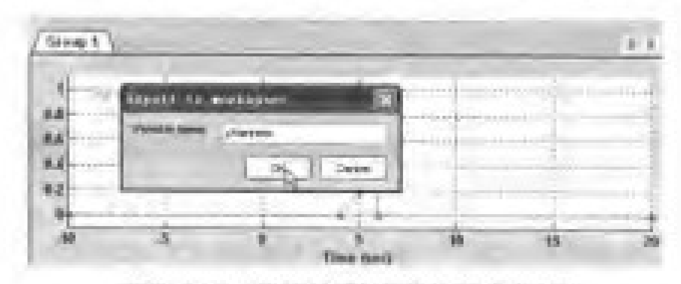


图 8.111 将波形输出到工作空间中

step 10 添加新的信号。选择菜单栏中的 "Signal" ⇨ "New" ⇨ "Sampled Gaussion noise" 命令，打开 "Add sampled Gaussion noise" 对话框，在其中设置 Gaussion 样本信号的频率、平均值和标准方差的数值，如图 8.112 所示。



图 8.112　添加新的信号对象

step 11 查看添加的信号。当单击"Add sampled Gaussion noise"对话框中的"OK"按钮后，就可以在该编辑器中添加"Signal 2"信号，如图 8.113 所示。

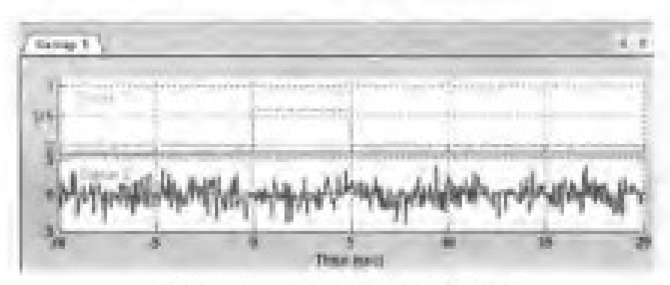


图 8.113　添加后的信号对象

step 12 删除原始的信号对象。在信号列表中选择"Signal 1"，然后选择菜单栏中的"Edit"⇨"Delete"命令，删除原来的信号对象，如图 8.114 所示。

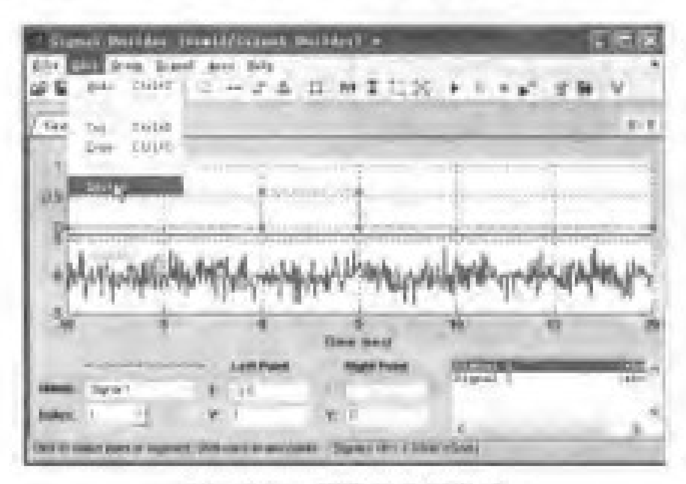


图 8.114　删除原来的信号

step 13 查看删除后的信号对象。选择相应的菜单选项后，系统将原来的信号删除，得到如图 8.115 所示的结果。

step 14 设置仿真参数。选择菜单栏中的"File"⇨"Simulation options"命令，打开"Simulation Options"对话框，在其中设置仿真的参数，如图 8.116 所示。

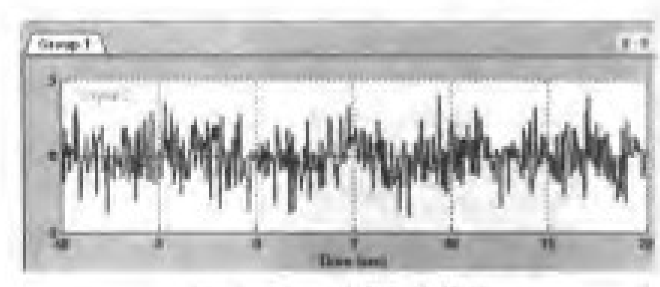


图 8.115 删除后的信号

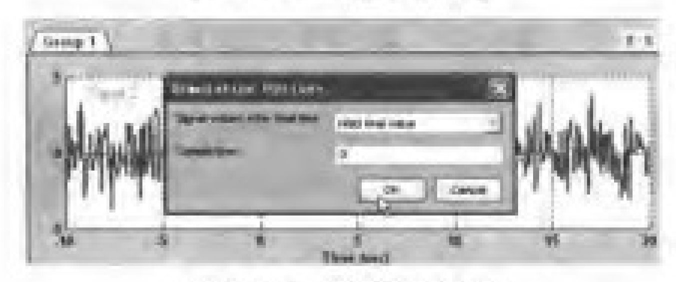


图 8.116 设置仿真参数

在该对话框中，将 "Sample time" 设置为 0，表示 Simulink 的输出结果是连续的图形，这也是系统的默认设置。单击对话框中的 "OK" 按钮，完成设置。

step 15 查看仿真结果。返回模型设计界面，单击 "开始仿真" 按钮，并双击 "Scope" 模块，查看仿真结果如图 8.117 所示。

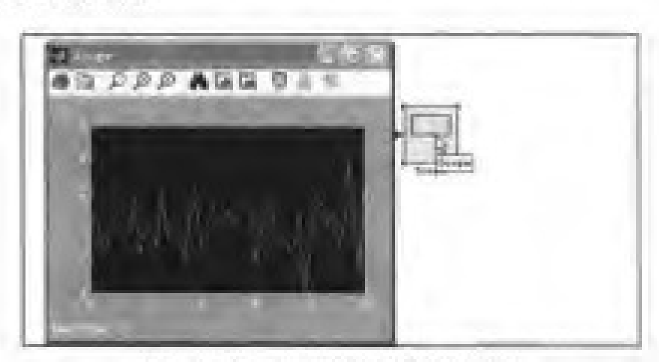


图 8.117 查看连续的仿真结果

step 16 修改仿真参数。返回到 Signal Builder 对话框中，选择 "File" ⇨ "Simulation options" 命令，打开 "Simulation Options" 对话框，修改仿真参数，如图 8.118 所示。

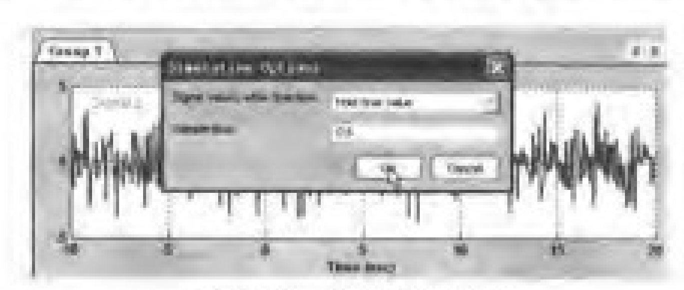


图 8.118 修改仿真参数

step 17 查看仿真结果。返回模型设计界面，单击 "开始仿真" 按钮，并双击 "Scope" 模块，查看仿真结果如图 8.119 所示。

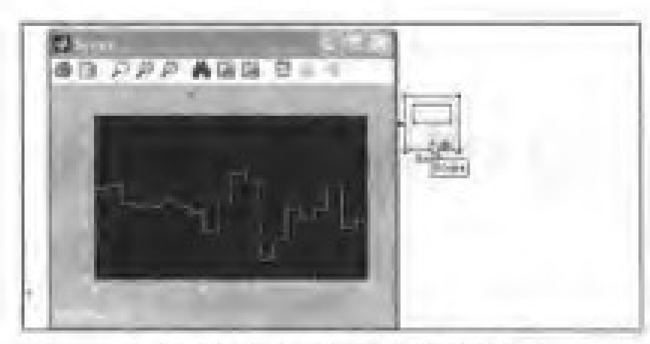


图 8.119　显示离散的仿真结果



Simulink还为"信号组"提供其他多种其他的信号类型，例如，Possion random noise、Triangle、Sampled sin等常见信号，用户可以自行了解这些信号的特征。

8.4.9 使用自定义信号源

除了使用前面章节所介绍的常用信号之外，还可以使用自定义的信号源创建仿真系统。在Source模块库中提供两种常用的模块"From File"和"From Workspace"，来使用用户自行定义的信号源。其中，"From File"模块将从MAT文件中获取信号矩阵，信号以行的方式存放，第一行表示时间变量，其余每行存放的是信号序列；"From Workspace"模块将从MATLAB的工作空间中指定的数组或者构架中读取数据。

在"From Workspace"模块中，其"Data"参数指定工作空间中的某个变量或者表达式，该变量或者表达式代表了一个二维数据矩阵或者是一个包含了信号数值和时间数值的构架变量。"From Workspace"模块的图标显示表达式中的"Data"参数数值。下面使用一个简单的例子来说明如何使用该模块创建自定义信号。

例 8.14　在Simulink中在$[0, 4\pi]$范围内绘制函数$y = e^{-\frac{t}{3}}\cos(\frac{t}{2})$的图形。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function Ts=mysource
t=linspace(0,4*pi,100);
y=exp(-t/3).*cos(1/2*t);
Ts=[ t',y'];
```

在输入上面的程序代码后，将其保存为"mysource.m"文件。

step 2 打开一个新的模型界面窗口，向其中添加模块，得到的结果如图8.120所示。

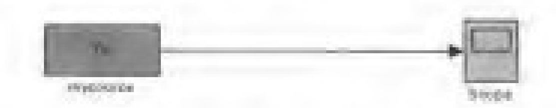


图 8.120　添加程序模块

step 3 设置"From Workspace"模块的参数。双击上面的"From Workspace"模块，打开对应的参数对话框，在"Data"数值框中输入"Ts"，如图8.121所示。

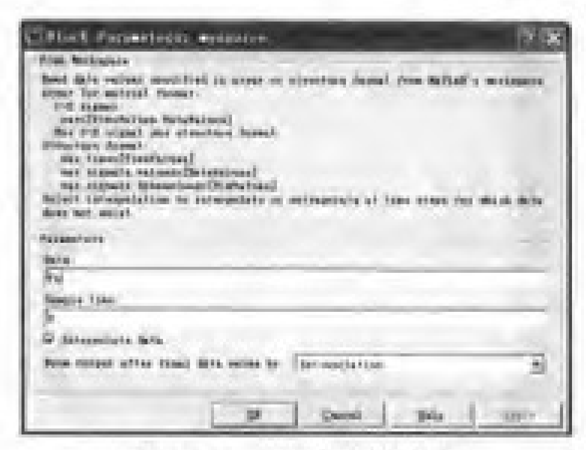


图 8.121　设置模块的参数

step 4 在 MATLAB 的命令窗口中输入下面的命令：

```
>> Ts=mysource;
```

上面程序代码在 MATLAB 的工作空间中添加 Ts 变量。

step 5 返回到模型界面中，设置仿真时间为 "12"，单击 "开始仿真" 按钮，进行系统的仿真，并双击 "Scope" 模块，查看仿真结果，如图 8.122 所示。



图 8.122　查看仿真的结果

step 6 在命令窗口中演示仿真结果。为了确定仿真的结果，可以在 MATLAB 的命令窗口中绘制相应的函数图形，在命令窗口中输入下面的代码：

```
>> t=linspace(0,4*pi,100);y=exp(-t/3).*cos(1/2*t);
>>plot(t,y,'r','LineWidth',2)
>>grid; axis tight
```

step 7 查看图形结果。输入代码后，按 "Enter" 键，得到的图形如图 8.123 所示。

8.4.10 信号接收器

在前面的章节中，读者已经多次接触到各种 Simulink 仿真系统的模块结构。其中常用的信号接收器包括 Scope、Display 和 Terminator 等模块，可以根据需要选中合适的信号接收器。由于 Scope（示波器）模块是最常见的信号接收器模块，因此，在本小节中将详细介绍关于 Scope 模块的基础知识。

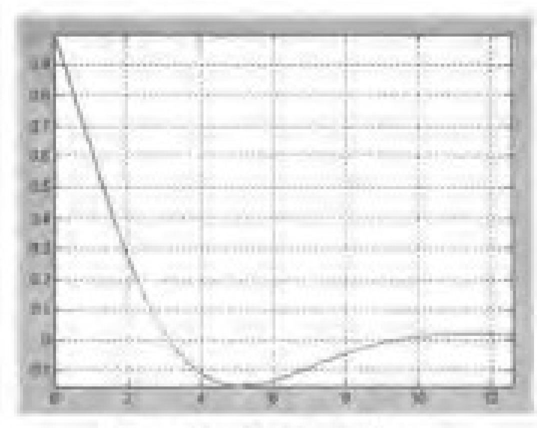


图 8.123　命令窗口的结果

在 Simulink 中，示波器的用途主要用来接收向量信号，在仿真过程中，示波器实时显示信号波形，但是该波形不能被直接打印或者嵌入文件中。无论示波器是否打开，只要用户运行仿真，示波器的缓冲区就会接收送来的信号，该缓冲区中最多可以接收 30 多个信号，它们以列的方式排列。

下面将以例 8.14 中的示波器为例，说明如何编辑示波器。

例 8.15　演示如何编辑示波器的属性。

step 1 设置纵坐标的属性。选中上面案例中的示波器，然后单击鼠标右键，在弹出的快捷菜单中选择 "Axes Properties" 菜单选项，如图 8.124 所示。



图 8.124　设置示波器的坐标属性

当选择相应的菜单选项后，Simulink 弹出相应的对话框，在对话框中设置 Y轴的上下限数值，如图 8.125 所示。



图 8.125　设置 Y 坐标轴的上下限

step 2 查看修改后的结果。单击上面对话框中的 "Apply" 按钮后，应用上面的属性设置，然后单击 "OK" 按钮，关闭对话框，查看修改后的示波器模块，如图 8.126 所示。

step 3 设置横坐标的属性。在上面的示波器界面中，单击 "Parameters" 按钮，打开 "示波器属性" 对话框，选择 "General" 选项卡，在其中设置坐标轴的属性，如图 8.127 所示。

图 8.126 修改后的显示情况

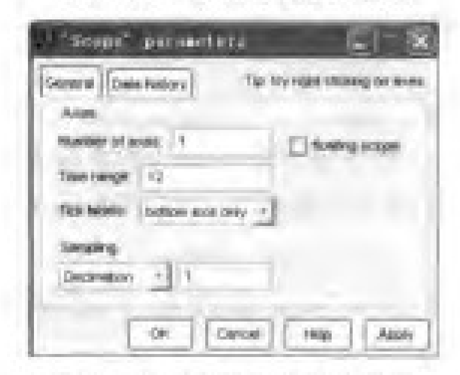


图 8.127 设置横坐标的属性

下面详细介绍上面选项卡的各选项含义。

- **"Time range" 选框：** 该选框设置的是横坐标轴的刻度范围，由于示波器显示的就是波形，因此横坐标标轴就是时间轴。"Time range" 选框的默认数值为 10，表示默认情况下 Simulink 显示的是区间[0，10]的波形。如果选择 "Autoscale" 按钮，则该选框将显示 "Auto"。可以根据需要修改其数值，例如在本例中选择的是 12。
- **"Sampling" 下拉菜单：** 该下拉菜单包含两个菜单选项：抽选 Decimation 和采样时间 Sample time。其中，Decimation 选项用来设置显示频度，如果选择 n，则每隔 n－1 个数据点给予显示，默认数值为 1；Sample time 选项则用来设置显示数据点的采样时间步长。默认数值为 0，表示显示连续信号；如果输入的数值为－1，则表示信号显示的方式取决于输入的信号；如果输入的数值大于 0，则显示离散信号。

当设置上面的属性后，选择对话框中的 "Data history" 选项卡，在其中设置关于信号数据的属性，如图 8.128 所示。

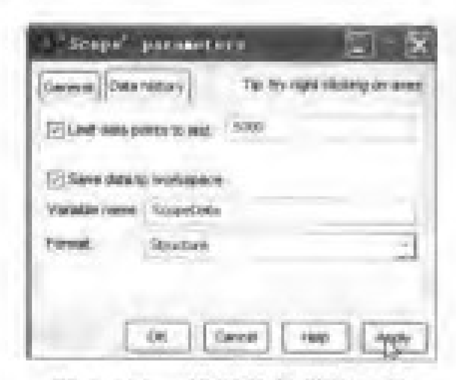


图 8.128 设置信号数据属性

下面详细介绍该选项卡中各选项的具体含义。

- **Limit datapoints to last**：设置缓冲区接收数据的长度。在默认情况下，系统会选中该选项，同时将其数值设置为5000。如果输入数据的长度大于这个数值，则更早的历史数据点会被系统清除。
- **Save data to workspace**：在默认情况下，该选项将不选中。如果选中该选项，则将缓冲区中的数据送到工作空间中。用户可以自行设定变量名称。



上面对话框中列出的关于示波器的其他属性，限于篇幅这里就不详细介绍了，感兴趣的读者请自行尝试。

在本小节的最后，将介绍关于示波器的另外一个重要属性——游离示波器。在示波器的"General"面板中选中"floating scope"选项，将Scope模块转换为游离示波器。游离示波器是可以显示多个信号的Scope信号模块，下面以一个简单的实例来演示如何在Simulink中使用游离示波器。

例 8.16 演示如何在Simulink中使用游离示波器。

step 1 打开空白模型窗口界面，向界面中添加各模块，得到的结果如图8.129所示。

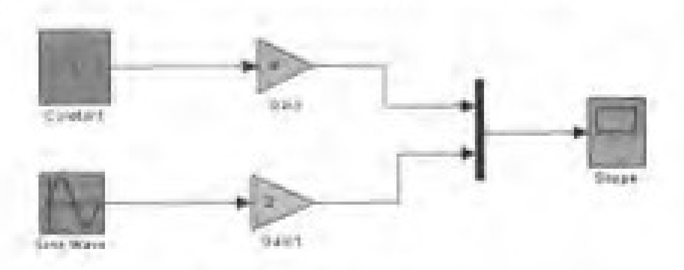


图 8.129 系统模块

step 2 将仿真的时间设置为"20"，单击"开始仿真"按钮，进行仿真，然后双击上面的"Scope"模块，查看仿真结果，如图8.130所示。

step 3 设置"Scope"模块的参数。打开Scope模块的对话框，将其模块的坐标轴个数设置为"2"，如图8.131所示。

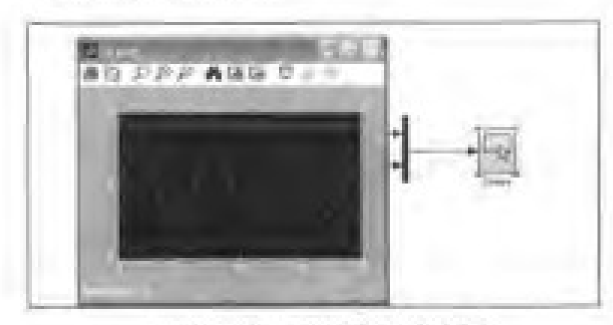


图 8.130 查看仿真的结果



图 8.131 设置坐标轴个数

step 4 返回到模块编辑界面，重新添加新的模块连接。由于上面步骤添加了一个坐标轴，因此在模块编辑界面中"Scope"模块会多出一个连接端口，将其和"Mux"模块连接起来，得到的结果如图8.132所示。

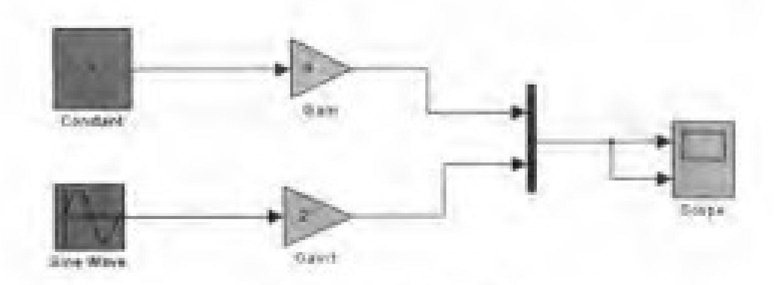


图 8.132 重新连接模块

step 5 重新查看仿真结果。将仿真的时间设置为“20”，单击“开始仿真”按钮，进行仿真，然后双击上面的“Scope”模块，查看仿真结果，如图 8.133 所示。

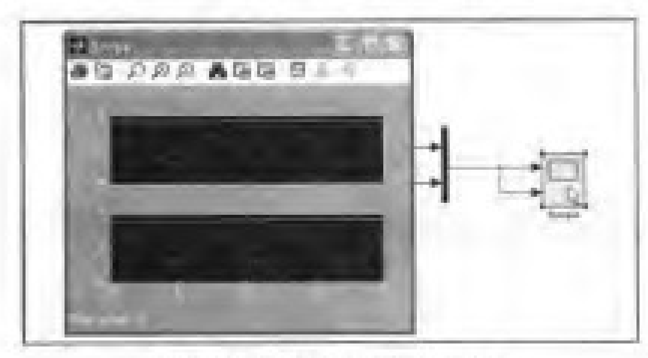


图 8.133 查看新的仿真结果

step 6 将示波器转换为“游离示波器”。单击 Scope 模块中的“Floating scope”按钮，将示波器转换为“游离示波器”，如图 8.134 所示。



图 8.134 转换为游离示波器

step 7 打开“Signal Selector”对话框。选中对话框中的第一个坐标系，然后单击鼠标右键，在弹出的快捷菜单中选择“Signal Selection”选项，如图 8.135 所示。



图 8.135 选择“Signal selection”选项

当选择 "Signal Selection" 选项后，Simulink 会打开 "Signal Selector" 对话框，在其中选择第一个坐标系统的显示波形，如图 8.136 所示。



图 8.136　选择第一个坐标轴中的波形

step 8 设置坐标轴的显示属性。在上面对话框中的 "List contents" 下拉菜单中选择 "Selected signals only" 选项，如图 8.137 所示。

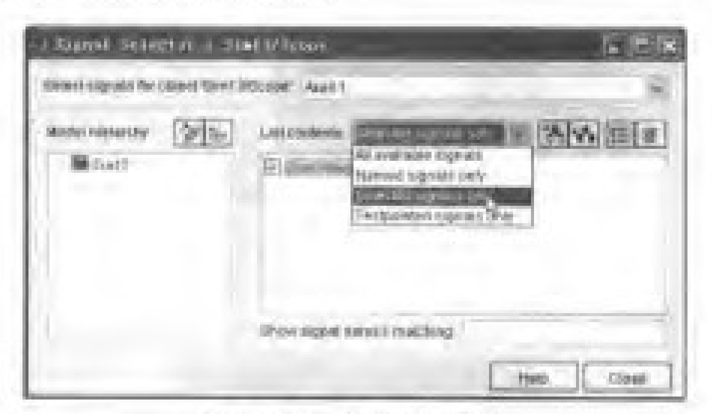


图 8.137　设置显示选项

step 9 重新查看仿真结果。将仿真的时间设置为 "20"，然后单击 "开始仿真" 按钮，进行仿真，然后双击上面的 "Scope" 模块，查看仿真结果，如图 8.138 所示。

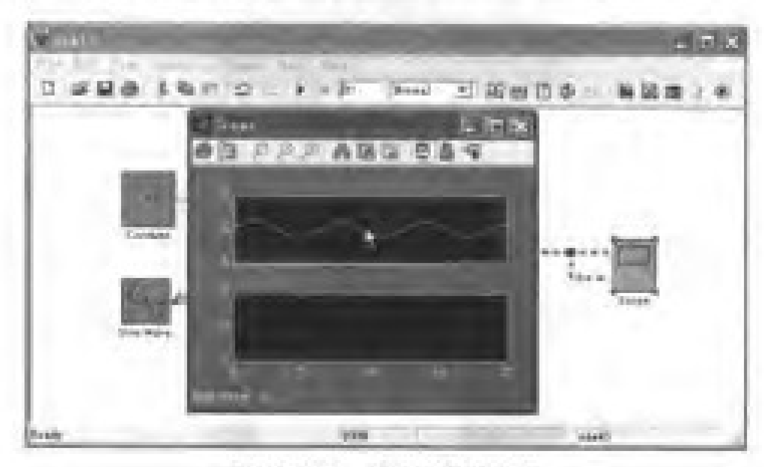


图 8.138　新的仿真结果



从上面的结果中可以看出，当设置只在"Axes1"中显示"Sine Wave"波形之后，重新运行仿真，在第坐标轴中只显示正弦波形。

step 10 重复上面的步骤，将"Axes 2"坐标系统设置为只显示常数的波形，其对应的"Signal Selector"参数对话框如图 8.139 所示。



图 8.139　设置"Axes 2"的信号属性

step 11 查看仿真结果。在设置信号属性后，重新运行仿真，得到的结果如图 8.140 所示。

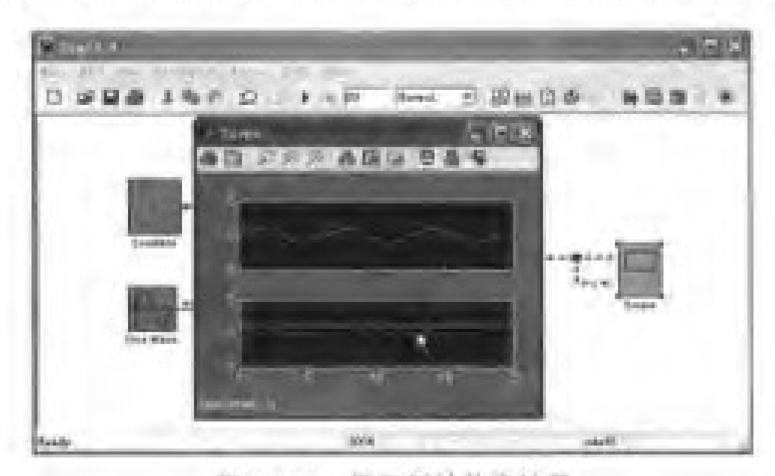


图 8.140　显示新的仿真结果

8.5 Simulink 仿真的设置

根据前面章节的介绍，Simulink 模型在本质上就是一组计算机程序，它定义描写仿真系统的一组微分或者差分方程。当开始整个系统的仿真后，Simulink 就开始使用一种数值求解方法来求解方程。在进行系统仿真之前，如果不希望采用系统的默认参数设置，就必须在运行仿真之前设置各种仿真参数。

在 Simulink 中，需要设置的仿真参数主要包括：起始时间和终止时间、仿真步长、仿真容差、数值积分算法等，还可以设置系统是否从外界获得数据、是否向外界传递数据等仿真参数。所有这些参数都可以在"Configuration Parameters"对话框中完成设置。下面分小节来详细介绍如何设置各种参数。

8.5.1 设置解算器参数

在设置仿真参数之前，首先需要打开"Configuration Parameters"对话框，选择模型窗口中的

"Simulation" ⇨ "Configuration Parameters" 命令，打开对话框，如图 8.141 所示。



图 8.141　仿真参数对话框

首先介绍如何设置解算器的参数值。在"Configuration Parameters"对话框的左侧选择"Slover"选项，对应的 Slover 面板如图 8.142 所示。

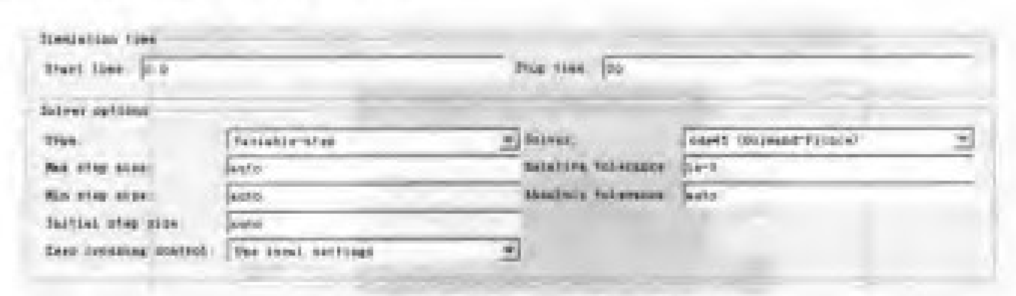


图 8.142　解算器参数设置面板

下面将详细介绍上面对话框的各项参数的含义。

- **Simulation time面板**：可以在该面板中设置系统仿真的起始时间和终止时间，在默认情况下，系统的起始时间设置为 0，终止时间为 10。可以在 "Start time" 选框中设置新的起始时间，在 "Stop time" 选框中设置新的终止时间。



可以直接在模块操作界面中设定终止时间，但是不能直接在该界面设定系统仿真的起始时间。

- **Solver options 面板**：可以在该面板中设置解算器的各种属性。其中，在 "Type" 下拉菜单中可以选择解算器的类型。在 Simulink 中，提供固定步长（Fixed-Step）解算器和变步长（Variable-Step）解算器两类解算器，默认设置是变步长的 ode45。两种解算器在计算下一个仿真时间的方法，都是在当前仿真时间上加上一个时间步长。其主要区别在于，固定步长计算器的时间步长是常数，而变步长解算器的时间步长是根据模型动态特征来变化的。



当系统模型的状态变化比较快时，为了保证计算精度，必须降低时间步长，相反，如果模型的状态变换比较慢时，则有必要增加时间步长。

这里根据笔者的经验，给出选择不同类型的计算器的方法。当希望通过自建模型生成代码并在实时计算系统中运行这些代码时，可以选择固定步长解算器来仿真模型，这样做的原因在于实时计算系统会以固定的采样速率运行，若采用变步长将有可能使仿真发生错误。

当不希望从模型生成代码时，解算器的选择将取决于模型的动态特征。当模型的状态变化特别快或者包含不连续状态时，选择可变步长解算器可以缩短仿真时间。这是因为可变步长相对于固定步长的解算器需要更少的时间，而且两者的计算精度相当。

当在 "Type" 下拉菜单中选择不同类型的解算器时，在其右侧的 "Slover" 下拉菜单中会显示不同类型的解算器列表，表 8.6 列出了 Simulink 中的定步长常见解算器。

表 8.6　Simulink 中的定步长解算器

解算指令 Slover	使用的方法
ode5	ode45 的定步长的计算形式
ode4	采用定步长的经典 4 阶的 Runge-Kutta 计算方法
ode3	ode23 的定步长的计算形式
ode2	定步长的 2 阶的 Runge-Kutta 计算方法
ode1	定步长的 Euler 方法
discrete	纯离散系统的特殊计算方法



关于变步长的解算方法的具体 Slover 种类，限于篇幅这里就不详细展开介绍了，感兴趣的读者请自行查看相关的帮助文件。

最后将介绍默认采用的ode45计算方法特点，在Simulink中，默认的Slover类型是变步长的ode45方法。之所以选择这些方法作为默认方法，是因为这种解算器能在保证计算精度的前提下，使用尽可能大的步长，能够排除积分步长和输出 "解点" 间隔之间的相互制约，可以不必为获得光滑输出而设定很小的步长。



关于该面板其他的属性，例如变步长解算器的步长、容差和输出设置等，都将在后面的章节中详细介绍，这里就不展开分析了。

8.5.2　仿真数据的输入输出设置

在Simulink中，设置向MATLAB工作空间中输出模型仿真结果数据或者设置从工作空间中读入仿真结果，也是属性设置的一个重要内容。为了能够设置这方面的属性，需要在 "Configuration Parameters" 对话框的左侧选择 "Data Import/Export" 选项，在其右侧就会显示 "Data Import/Export" 面板，如图 8.143 所示。

在详细介绍该设置页的功能时，首先有必要介绍 Simulink 的内状态向量的相关知识。前面已经介绍过，Simulink模型可以认为是一组联立的一阶微分或者差分方程。构成模型的传递函数模块、状态方程模块、非线性模块等都会随着相应的状态变量，因此就会引出状态变量的存放（Access）问题，在 Simulink 中解决存放问题的方法就是设置仿真数据的输入或者输出的属性。



Simulink 中提供直接获取系统的状态向量的具体指令，用户可以从中获得详细的状态信息，这些内容都在后面章节中详细介绍。

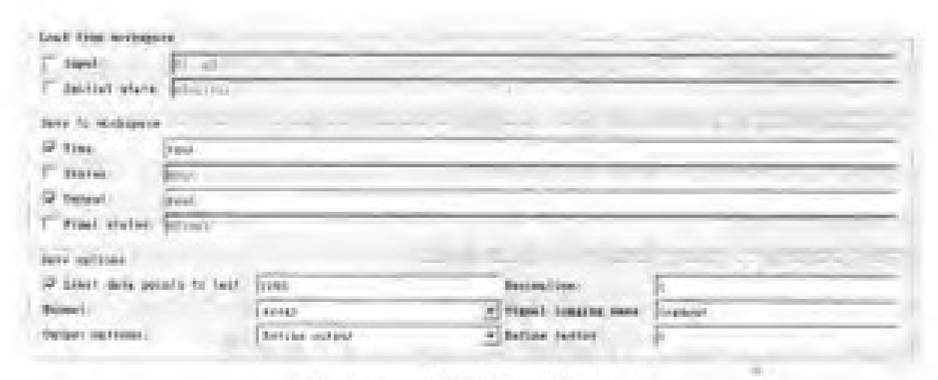


图 8.143 数据输入 / 输出设置

下面详细介绍该面板各选项的具体含义。

✦ Load from workspace 选项区。该选项区主要包括 "Input" 和 "Initial state" 两个选项框，这些选项框的主要内容如下。

- **Input 选项框**：如果在 Simulink 系统中选用输入模块 In，则必须选中该选项，并需要填写在 MATLAB 工作空间中的输入数据变量名称，例如[t, u]或者 TU。如果输入模块中有 n 个，则 u 的第 1、2、3…n 列分别送往输入模块 In1、In2…In n。
- **Initial state 选项框**：如果选中该选项框，则强迫系统模型从工作空间中获取模型中所有内状态变量的初始数值，而不管构成该模型的积分块是否设置过怎样的初始数值。该选项框所填写的变量名称应该是工作空间中所存在的变量。

✦ Save to workspace 选项区。该选项区主要包括 Time、States、Output 和 Final state 选项框，下面详细介绍这些选项框的具体含义。

- **Time 选项框**：如果选中该选项框，系统模型就会将时间独立变量以指定的变量名称（默认名称为 tout）存放在工作空间中。
- **States 选项框**：如果选中该选项框，系统模型将会将其状态变量以指定的变量名称（默认情况下为 xout）存放在工作空间中。
- **Output 选项框**：如果系统模型中使用输出模块 Out，就必须选中该选项框，并填写在 MATLAB 工作空间中的输出数据变量名称。数据的存放方式与输入情况类似。
- **Final state 选项框**：如果选中该选项框，系统将会向工作空间中以指定的名称（默认名称为 xFinal）存放最终状态值。如果最终状态向量在该模型的新一轮仿真中又被用作初始值，那么这新一轮仿真是前一轮仿真的 "继续"。



在默认情况下，Simulink 会选中上面对话框中的 "Time" 和 "Output" 选项，用户可以根据需要来选择所需的选项。

✦ Save options 选项区。该区的主要功能是用来设置系统结果的保存属性。需要特别注意的是，该区的各种属性必须和 Save to workspace 区中的相应属性配合使用。下面详细介绍该区的各种选项的内容。

- **Limit rows to last 选项框**：如果选中该选项框，可以设定保存变量接收数据的长度，默认数值为 1000。如果输入数据长度超过设定的数值，则 "历史" 数据会被清除。
- **Decimation 选项框**：设定 "解点" 的保存频度。如果选择的是 n，则每隔 n－1 个数据点

就会保存一个"解点"。默认数值为 1。

- **Format 选项框**：Simulink 提供数组、构架和带时间的构架三种保存数据的格式。

8.5.3 仿真诊断设置

在 Simulink 中，提供多种异常情况的诊断属性设置，并将其分为多个子目录详细列出。其中，关于"Slover"的异常诊断属性如图 8.144 所示。

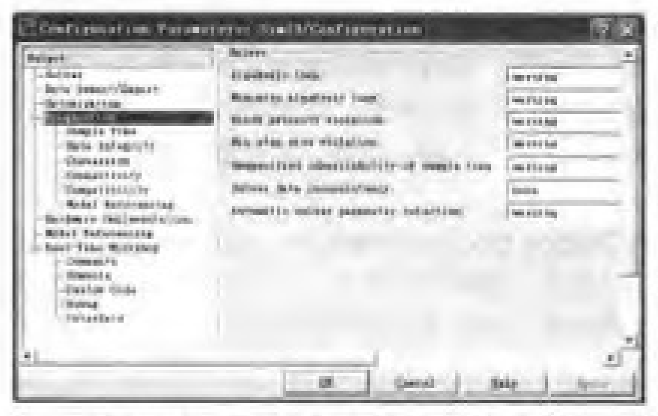


图 8.144　关于"Slover"的异常诊断

从上面的对话框中可以看出，在"Diagnostics"目录下有多个子目录，分别表示其他类别的异常诊断属性。例如，关于数据兼容性的诊断属性页面如图 8.145 所示。



图 8.145　关于"Data Integirty"的异常诊断

在详细介绍诊断情况之前，先介绍 Simulink 中关于诊断的几种处理方法。

- **None**：提示 Simulink 可以在仿真过程中忽略这种异常。
- **Warning**：提示 Simulink 每当遇到这种异常时发出相应的警告信息。
- **Error**：提示 Simulink 采用发布出错信息并终止仿真的方式来处理该异常情况。

关于异常诊断的情况比较复杂，本小节中将只列出几种比较常见的诊断选项。

- **Algebraic loop**：代数环的异常处理。在 Simulink 的系统模块中，如果存在代数环的异常，将会大大减慢仿真速度，甚至会导致仿真失败。对于这种异常，Simulink 通常采用 Warning 的处理方式。如果已知代数环的存在，而系统的仿真性能还可以接收，则可以将异常处理方式转换为 None。
- **Min step size violation**：用来处理最小步长欠小的异常情况。如果发生这样的异常情况，表

明微分方程解算器为了达到指定精度需要更小的步长，但是这是解算器所不允许的。解决的方法是采用更高阶的解算器来放松对步长的要求。对于这种异常情况，Simulink通常采用的方式是 Warning 或者 Error 处理方式。

- **Unconnected block input**：模块输入悬垂。这种异常情况表示在构架模块的时候有没有被使用的输入端。造成这种情况通常是因为疏忽，如果造成这种状态有其他的用途，建议将该输入端与"接地模块"相连，处理这种异常情况的主要方式是 Warning 或者 Error。
- **Unconnected block output**：模块输出悬垂。这种异常情况表示在构架模块的时候有没有被使用的输出端。造成这种情况通常是因为疏忽，如果造成这种状态有其他的用途，建议将该输出端与"终端模块"相连，处理这种异常情况的主要方式是 Warning 或者 Error。
- **Consistency checking**：一致性检验。这种为了专门调试用户自制模块的正确性，对于 Simulink 的标准模块不必进行一致性检验，因此通常该选项会选择"None"，以免影响这个系统的仿真速度。
- **Invalid root Inport/Outport block connection**：该选项用来处理其他模块和根目录级别的输出模块的异常连接的情况。选择该选项后，Simulink会在检测程序代码过程中，发现任何和输出模块中不正常的连接，则会当作异常进行处理。



上面列出的几种常见的异常处理方式并不是确定的，除了可以了解Simulink中的常见错误之外，还可以利用诊断来起到各种作用，在后面的章节中将详细介绍。

8.6 Simulink 线性系统建模

在 Simulink 中，连续系统是指可以使用微分方程来描述的系统。在很多领域中，例如物理等都是连续时间的，连续时间系统又可以分为线性和非线性两类。Simulink中提供用户创建连续系统的主要模块包括 Continuous、Math 和 Nonlinear 模块等。

连续系统模型在现实的各个领域中有着广泛的应用，所涉及的模块也比较繁多。本节选择几个比较常见的模块，介绍如何创建连续系统模块。

8.6.1 线性系统建模简介

相对于非线性系统而言，线性系统比较简单，所涉及的模块也比较容易。因此，在本小节中将以几个简单的实例来说明如何创建线性系统。

例 8.17 创建一个自行重新设置（Self-Resetting）的积分系统。

step 1 打开新的模块界面，然后向其中添加基本的积分模块，如图 8.146 所示。

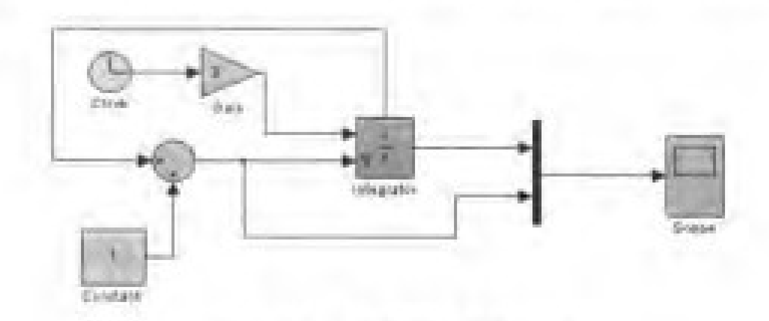


图 8.146 添加基本系统模块

step 2 设置"Sum"模块的属性。双击上面系统中的"Sum"模块，打开其对应的模块属性对话框，在"List of signs"选框中设置修改为"|-+"，如图 8.147 所示。

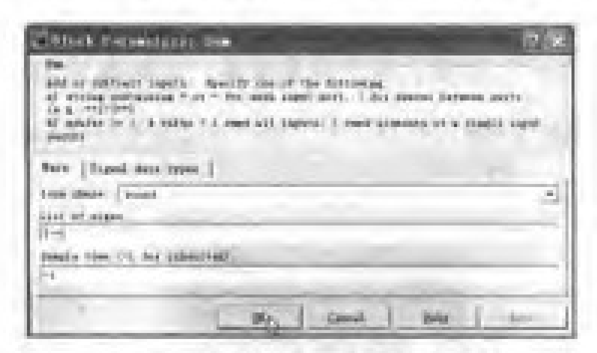


图 8.147　设置"Sum"模块的属性

在Simulink 中，"Sum"模块在"Math Operations"模块库中，其主要功能是完成输入变量的相加或者相减的数学运算。一般情况下，其输入变量可以是标量、向量或者矩阵。可以在"List of signs"区域中输入相应的数学运算符，例如，加号、减号或者分隔号等，在默认情况下其数值为"|++"，表示两个输入变量的相加运算。可以根据情况修改输入变量的个数和运算关系，在本实例中设置为"|-+"，修改后的模块如图 8.148 所示。

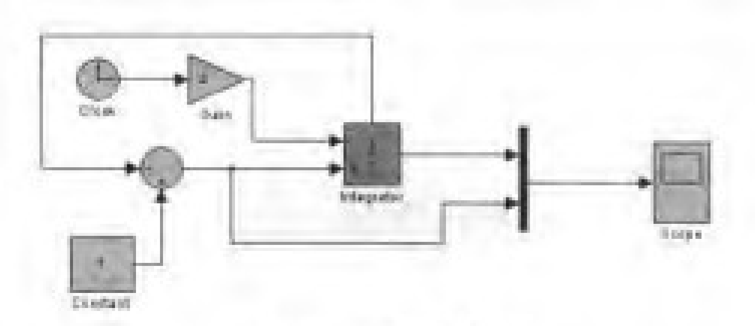


图 8.148　修改后的模块

step 3 设置"Integrator"模块的属性。双击上面系统模块中的"Integrator"模块，打开对应的属性对话框，设置对应的属性，如图 8.149 所示。



图 8.149　设置"Integrator"模块的属性

在对话框的"External reset"选框中选择"falling"选项，表示此端口的信号由正变成负

的瞬间，积分器会被强迫归零。同时，选中对话框中的 "Show state port" 选项。这样，积分器模块就会添加一个输出端口，该端口将会输出关于积分状态的信号。

step 4 运行仿真系统。完成属性设置以后，将模块的仿真时间设置为 6，然后选择模块界面中的 "Simulation" ⇨ "Start" 命令，得到仿真结果，如图 8.150 所示。



图 8.150　仿真得到的结果

在上面的仿真系统中，之所以将仿真时间设置为6，是因为在本仿真结果中在该时间范围内的结果是比较明显的。

step 5 修改纵坐标的数值。如果延用系统默认的 10 秒，无法显示其真正的仿真结果图形，为了更好地查看仿真的结果，可以设置仿真纵坐标的数值，如图 8.151 所示。



图 8.151　设置系统仿真的显示范围

step 6 查看新的仿真结果。设置信号显示范围后，单击图 8.151 对话框中的 "Apply" 按钮后，再单击 "OK" 按钮后，得到新的仿真结果如图 8.152 所示。

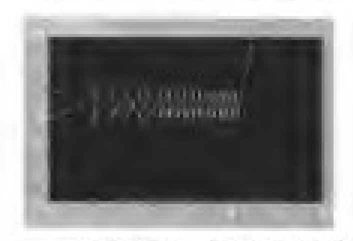


图 8.152　新的仿真结果

在本实例完成之后，需要分析该整个系统的基本原理。

step 1 首先，由 "Clock" 模块产生系统仿真时间变量，然后经过增益模块 "Gain" 得到结果变量 $2t$，再将该变量经过积分器模块得到输出变量 $f(t)=\int_0^t 2t\mathrm{d}t=t^2$，将该积分结果输入到示波器模块 "Scope" 中，这就是上面仿真系统中的中间线路。

step 2 同时，在上面仿真系统中还包括"自设置"模块。首先，通过"积分器"模块的"State port"的输出端输出系统仿真的状态量 t^2，然后通过"Sum"模块将该状态变量和常数 1 相连，得到"积分器"模块的状态重设端口信号变量"$1-t^2$"。

step 3 由于在前面步骤中将"External reset"选框中选择"falling"选项，表示此端口的信号由正变成负的瞬间，积分器会被强迫归零。也就是说，当 $1-t^2=0$ 时，即当 $t=1$ 时，积分器的初始值被设置为 0。而在此时间之后的积分则变为 $f(t)=\int_1^t 2t dt=t^2-1$，因此当该积分计算式再次达到上限积分数值 1 时，即当 $1-(t_2^2-1)=0 \Rightarrow t_2=\sqrt{2}$ 时，积分器再次将积分的初始数值设置为 0，而此后的积分数值变为 $f(t)=\int_{\sqrt{2}}^t 2t dt=t^2-2$，后面情况依次类推。

step 4 上面分析的情况就是"Scope"模块得到的第一维向量的函数表达式。对于上面对话框中的第二维向量表达式，根据上面系统约束的条件，其满足 $g(t)=1-f(t)$。

8.6.2 线性系统建模实例

在本小节中，将再以一个具体的实例来演示如何在 MATLAB 中创建线性系统建模。

例 8.18 创建一个 Simulink 系统，演示向上抛投小球的运动轨迹。

step 1 打开新的模块界面，然后向其中添加基本的积分模块，如图 8.153 所示。

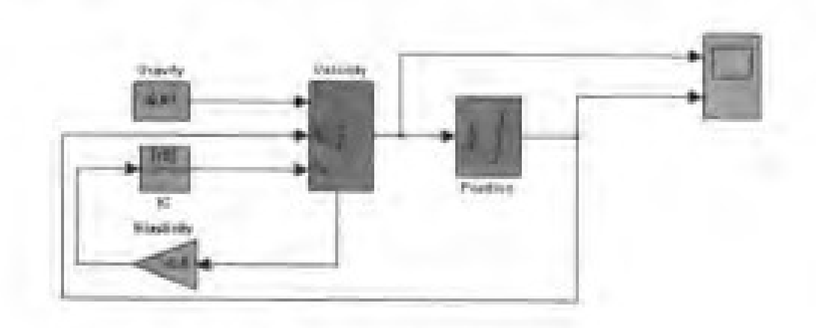


图 8.153 添加基本系统模块

step 2 设置第一个积分器的属性。双击左侧第一个积分器，打开积分器的属性对话框，在其中设置其属性，如图 8.154 所示。

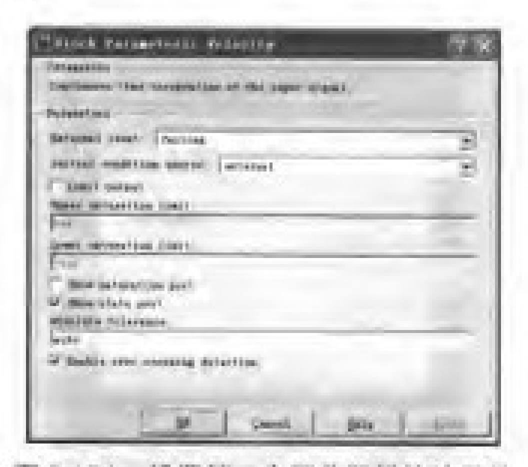


图 8.154 设置第一个积分器模块的属性

在上面的系统模块中，第一个积分器的功能是积分得到小球抛投的速度，为了能够更加逼真

地模拟该抛投运动，需要为该积分器进行外部初始条件的设置，并为其设置新的重设条件端口。因此需要选中所有的相关端口。

step 3 设置第二个积分器的属性。双击第二个积分器，打开积分器的属性对话框，在其中设置其属性，如图 8.155 所示。

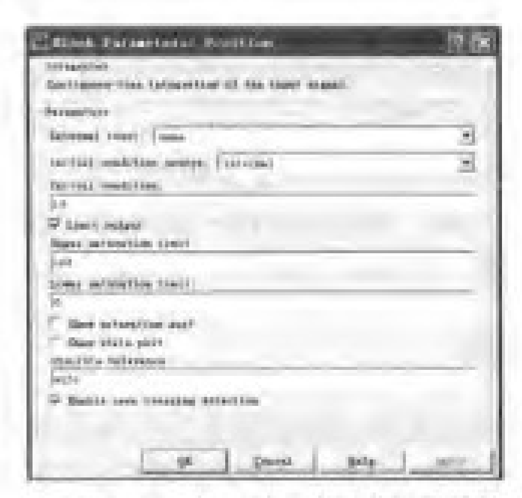


图 8.155　设置第二个积分器模块的属性

在上面的系统中，第二个积分器模块的功能是将速度积分得到小球运动的高度。因此，需要设置初始高度数值，在 "Initial condition" 选框中输入 "10"，同时选中 "Limit output" 选项，然后在 "Lower saturation limit" 选框输入积分下限数值 "0"，由于该积分器的输出变量是高度，所以其数值的下限为 0，其他选项保持默认设置。



在前面的步骤中，已经多次接触到 "Integrator" 模块的属性设置问题，根据不同的系统要求，需要设置不同的积分属性，关于这些属性设置问题将在后面的内容中详细介绍。

step 4 设置 "Scope" 模块的属性。根据本实例的要求，需要显示小球运动的高度和速度的图形，所以需要将其坐标系的个数设置为 2，如图 8.156 所示。



图 8.156　设置示波器的属性



之所以在上面的"Scope"中没有显示任何的波形，是因为在该步骤中还没有进行仿真工作。

step 5 查看仿真结果。将系统的仿真时间设置为20，然后选择模块界面中的"Simulation"⇨"Start"命令，得到仿真结果，如图 8.157 所示。



图 8.157 查看仿真结果

在上面的仿真结果中，上面的图形表示小球运动速度随着时间变化的曲线，大致符合线性关系；下面的图形则表示小球运动的高度随着时间变化的曲线，大致符合二次抛物线的关系。

step 6 简要分析该系统的原理。

本系统分析的是在初始高度为 10 米的地方以初速度 15m/s 向上抛投小球的运动轨迹，根据基础物理知识，选择重力加速度为 9.81 m/s^2。同时，考虑到空气阻力对小球运动的影响，每次进行积分的时候，将积分后的时间步（Time step）速度转换为前一个时间步的0.8倍，相当于用速度的减少来替代能量的损失，得到的结果就是包含衰减的小球运动轨迹图形和速度图形。



在上面系统模块中，用到了"Initial Condition"模块来设置积分的初始条件，该模块在连续时间，特别是积分器模块中会经常用到。关于该模块的使用方法，可以查看相应的帮助文件。

例 8.19 创建 Simulink 系统，求解二阶微分方程 $x''(t)+0.4x'(t)+0.9x(t)=0.7u(t)$的方程解，其中 $u(t)$是脉冲信号，需要使用 Simulink 来求解函数 $x(t)$。

step 1 改写微分方程。将上面需要求解的微分方程改为下面的形式：

$-0.4x'(t)-0.9x(t)+0.7u(t)=x''(t)$

step 2 使用 Simulink 来创建上面的微分方程，完成的系统模块如图 8.158 所示。

step 3 设置"Pulse Generator"模块的属性。双击上面系统模块中的"Pulse Generator"模块，打开对应的属性对话框，设置相应的模块属性，如图 8.159 所示。



在图 8.159 的对话框中，主要设置了 Pulse Generator"模块的信号频率。关于该信号的其他属性信息，请查看相应的帮助文件。

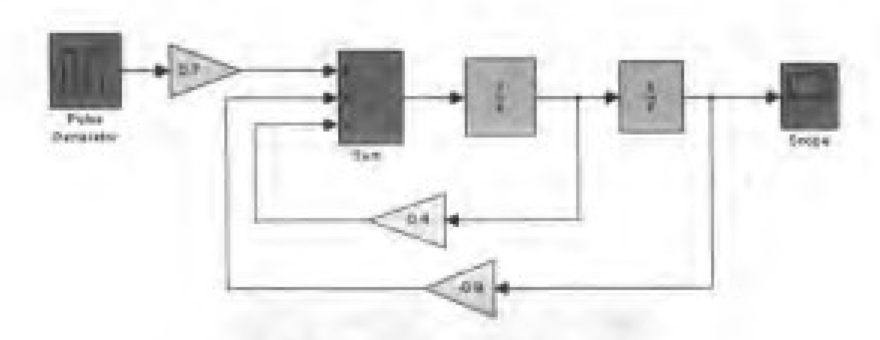


图 8.158　完成的系统模块



图 8.159　设置 "Pulse Generator" 模块的属性

step 4 设置 "Sum" 模块的属性。双击上面系统模块中的 "Sum" 模块，打开对应的属性对话框，设置相应的模块属性，如图 8.160 所示。

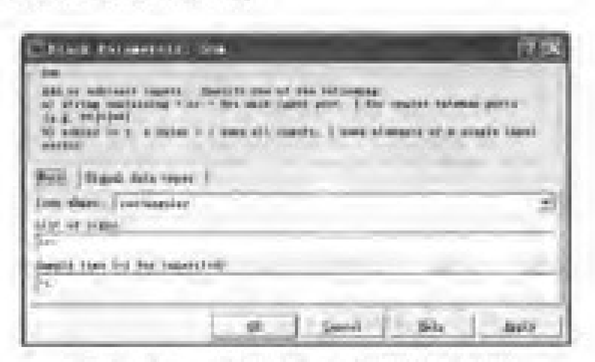


图 8.160　设置 "Sum" 模块的属性

在图 8.160 的对话框中，将模块的形状参数 "Icon shape" 属性设置为 "rectangular"，将模块设置为矩形，然后在 "List of signs" 选框中将符号设置为 "++-"。

step 5 查看仿真结果。将系统的仿真时间设置为10，然后选择模块界面中的"Simulation"⇨"Start"命令，得到仿真结果，如图 8.161 所示。

step 6 添加新的模块。图 8.161 仿真的结果传输到 MATLAB 的工作空间中。在上面仿真系统模块的基础上，添加 "Clock" 和 "To Workspace" 模块，将仿真的结果传输到工作空间中，如图 8.162 所示。



图8.161 仿真结果

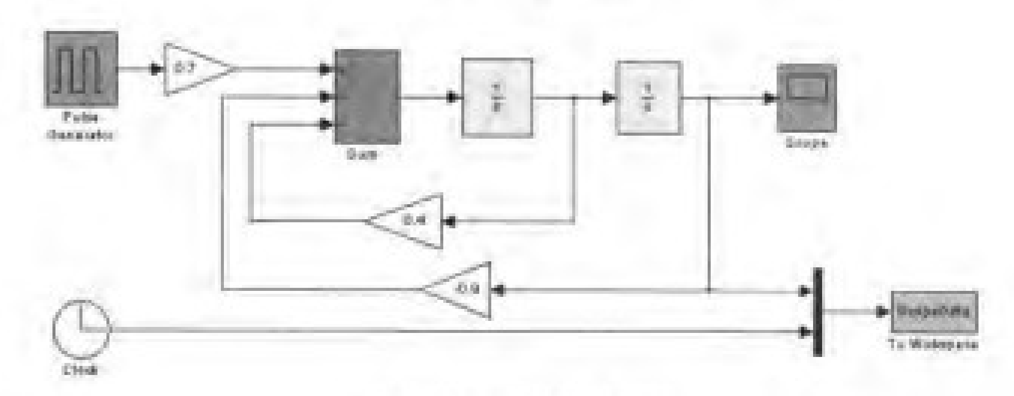


图8.162 添加新的模块

Clock模块的功能是产生系统仿真的时间变量t，在模块中将该时间变量和系统积分得到的$x(t)$，通过"To Workspace"模块传递给工作空间中的变量ScopeData。

step 7 设置"To Workspace"模块的属性。为了设置输出变量的属性，可以双击"To Workspace"模块，打开模块的属性对话框，在其中设置相应的属性，如图8.163所示。

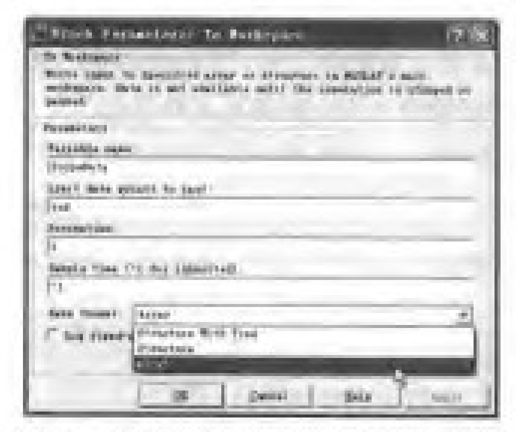


图8.163 设置"To Workspace"模块的属性

在图8.163的对话框中，将仿真结果的输出变量名称设置为"ScopeData"，同时将保存数据的格式设置为"Array"，即将仿真结果按照数组的格式输出数值结果。

step 8 处理输出数据。首先运行重新设置的仿真系统，然后返回到MATLAB的命令窗口中，输入下面的程序代码

```
>> clf;
>> t=ScopeData(:,2);
>> x=ScopeData(:,1);
>> [ xm, km] =max(x);
>> plot(t,x,'m','LineWidth',3),hold on
>> plot(t(km),xm,'y.','MarkerSize',24),hold off
```

```
>> grid
```

step 9 查看图形结果。在输入上面的程序代码后，按"Enter"键，得到的图形结果如图 8.164 所示。

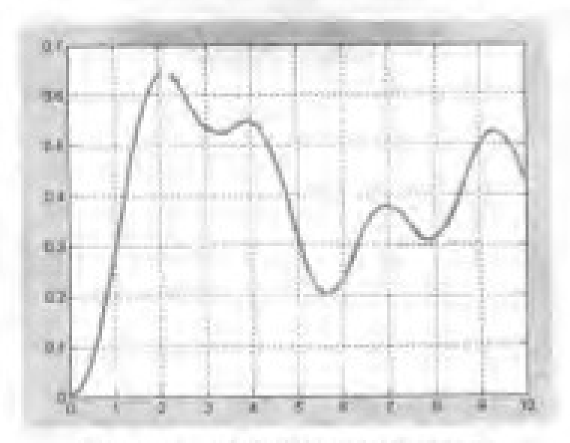


图 8.164　命令窗口中的绘制结果

从绘制图形结果可以看出，通过常用的 MATLAB 绘图命令得到的图形结果和仿真系统得到的图形完全相同，说明系统传递数据成功。实质上，在 SimuLink 中可以使用三种比较常见的方法来向 MATLAB 的工作空间中存放仿真数据：

- 使用示波器模块向 MATLAB 的工作空间中存放构架（Structure）数组；
- 使用"To Workspace"模块以选定的矩阵方式向工作空间存放数组；
- 设置仿真参数中的"Data Import/Export"选项的属性，以 tout、xout 的名称将数据存放在工作空间中。

在本实例中，使用的方法是上面的第二种，可以根据实际情况，使用不同的方法选择数据输出的方法。下面简单介绍使用上面介绍的第三种方法。

step 10 修改数据传递方法。选择模块编辑界面中的"Simulation"⇨"Configuration parameters"命令，打开"Configuration parameters"对话框，选择该对话框中的"Data Import/Export"选项卡，在对话框的左侧选中"Time"和"States"选项，如图 8.165 所示。

图 8.165　设置数据输出的选项

step 11 查看数据传递结果。当完成上面的设置以后，再次运行上面的仿真系统，查看 MATLAB 工作空间中的变量情况，得到的结果如下：

```
>> ScopeData
ScopeData =
          time: [ 61x1 double]
```

```
        signals: [1x1 struct]
      blockName: 'Sim19/To Workspace'
>> ScopeData.signals
ans =
          values: [61x2 double]
      dimensions: 2
           label: ''
>> ScopeData.signals.values
ans =
         0         0
    0.0000    0.0002
    0.0000    0.0012
    0.0000    0.0062
    0.0003    0.0313
    0.0084    0.1569
    0.0421    0.3569
    0.0986    0.5569
    0.1742    0.7569
    0.2648    0.9569
    0.3661    1.1569
    0.4157    1.2500
    0.5114    1.4500
    0.5821    1.6500
    ........................     //限于篇幅，这里省略了部分数据
    0.4508    8.7500
    0.4970    8.9500
    0.5225    9.1500
    0.5280    9.3500
    0.5149    9.5500
    0.4850    9.7500
    0.4407    9.9500
    0.4277   10.0000
```



从上面的结果可以看出，如果选择数据输出的"Time"和"States"选项，则可以通过内置的属性直接引用仿真数据，例如表达式ScopeData.signals.values就引用了仿真的时间和仿真数据。

例 8.20　使用传递函数模块，求解例 8.19 中的实例。

step 1 求解微分方程的传递函数。根据上面的实例中的微分方程：

$x''(t)+0.4x'(t)+0.9x(t)=0.7u(t)$

将上面的微分方程的两边同时进行变换，得到的结果如下：

$s^2X(s)+0.4sX(s)+0.9X(s)=0.7U(s)$

将上面的方程进行整理，得到转换公式如下：

$$G(s)=\frac{X(s)}{U(s)}=\frac{0.7}{s^2+0.4s+0.9}$$

step 2 根据上面的转换函数的公式，可以创建系统的模块，如图 8.166 所示。

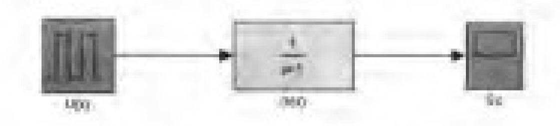


图 8.166　创建系统的模块

step 3 设置转换函数模块的属性。双击上面的转换函数模块，打开对应的对话框，在其中设置转换函数的公式，如图 8.167 所示。

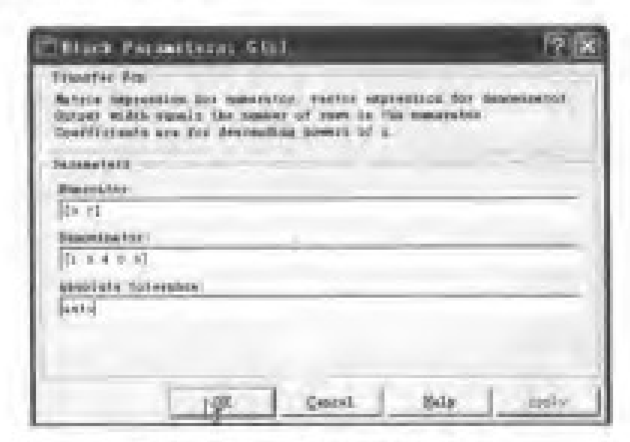


图 8.167　设置转换函数的属性

step 4 查看设置属性后的模块。当设置属性后，单击对话框中的 "OK" 按钮，得到的模块如图 8.168 所示。

step 5 查看仿真结果。完成上面的模块后，双击上面的 "Sc" 模块，查看仿真结果，得到的结果如图 8.169 所示。



图 8.168　设置属性后的模块



图 8.169　查看仿真结果



从上面的仿真结果可以看出，使用传递函数可以获得和原始系统模型一样的仿真结果，但是在结构上有了明显的改观。如果能够熟练使用传递函数，可以很方便地求解仿真模型。

例 8.21　使用状态方程模块来求解例 8.19 中的实例。

step 1 求解微分方程的状态方程。根据上面的实例中的状态方程：

假定 $x(1)=x, x(2)=x'$，则微分方程 $x''(t)+0.4x'(t)+0.9x(t)=0.7u(t)$ 可以转换为：

$$x'=\begin{bmatrix}x(1)\\x(2)\end{bmatrix}'=\begin{bmatrix}x(2)\\-0.9x(1)-0.4x(2)+0.7u(t)\end{bmatrix}=\begin{pmatrix}0 & 1\\-0.9 & -0.4\end{pmatrix}\begin{bmatrix}x(1)\\x(2)\end{bmatrix}+\begin{bmatrix}0\\0.7\end{bmatrix}u(t)$$

将上面的方程转换为下面的状态方程：

$$\begin{cases}x'=Ax+Bu\\y=Cx+Du\end{cases}$$

在上面的方程组中，参数 $A=\begin{pmatrix}0 & 1\\-0.9 & -0.4\end{pmatrix}$，$B=\begin{pmatrix}0\\0.7\end{pmatrix}$，$C=\begin{pmatrix}1\\0\end{pmatrix}$，$D=0$。

step 2 根据上面的状态方程，可以创建系统的模块，如图 8.170 所示。

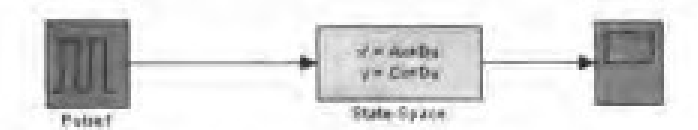


图 8.170　添加系统的模块



在 Simulink 中，状态方程模块是在 Continuous 模块库下的子模块，主要用来分析实施一个线性状态系统。

step 3 设置状态方程模块的属性。双击状态方程模块，打开对应的对话框，在其中设置转换函数的公式，如图 8.171 所示。



图 8.171　设置状态方程模块的属性

step 4 查看仿真结果。完成上面的模块后，双击 "Sc" 模块，查看仿真结果，得到的结果如图 8.172 所示。



图 8.172　查看仿真结果



在本小节中，使用多种方法来求解同一个微分方程，主要目的在于介绍使用 Simulink 中可以灵活使用各种模块完成建模工作。

8.6.3 "积分器" 模块的工作原理

根据上一小节的内容，在连续系统建模中 "Integrator" 模块是一个十分重要的模块，灵活应用 "Integrator" 模块可以完成各种复杂的连续系统。因此，熟悉 "Integrator" 模块各种属性对创建模

块是个十分重要的内容，在本小节中将详细介绍“Integrator”模块的各种常见属性的设置工作。

在Simulink中，“Integrator”模块的功能就是对信号进行积分，选择“Contiunous”模块库下的“Integrator”模块，然后添加到模型编辑界面中，得到的默认模块如图8.173所示。

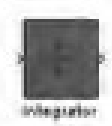


图8.173　默认的“Integrator”模块

“Integrator”模块输出其输入信号在当前时间步（time step）中的积分结果，可以使用一个简单的公式来描述其工作原理：

$$y(t)=\int_{t_0}^{t}u(t)\mathrm{d}t+y_0$$

在上面的公式中，$y(t)$表示的是“Integrator”模块的输出信号，也就是积分结果；$u(t)$表示“Integrator”模块的输入信号；y_0则表示积分的初始状态。其中，$y(t)$和$u(t)$都是当前仿真时间t的标量函数表达式。

在Simulink中，可以使用多种数值积分方法来计算“Integrator”模块的输出信号，每种方法各有优劣。在前面介绍过，在“Configuration Parameters”对话框的Slover面板中，可以选择不同的数值积分方法。Simulink将“Integrator”模块作为某个状态下的动态系统，其输入信号则是当前状态的时间导数值。下面的方程组可以表示该原理：

$$x=y(t)$$

$$x_0=y_0$$

$$x'=u(t)$$

除了“Integrator”模块的默认属性设置之外，“Integrator”模块还提供包含初始状态的积分模块，默认的初始状态值为0。使用“Integrator”模块的模块属性对话框，用户可以根据实际问题的需要设置新的初始状态数值，或者为“Integrator”模块添加初始状态数值端口。在“Integrator”的模块属性对话框中，一般可以设置：

- ◆ 设置积分的上限和下限；
- ◆ 添加一个输入端口来重设积分的初始状态数值；
- ◆ 创建一个可选的状态输出端口，状态输出端口一般用来引发仿真时间的重设。

8.6.4 设置初始状态数值

除了上面的设置工作之外，在模块的属性对话框中，还可以设置关于积分的其他各种属性，下面将介绍设置积分的初始状态数值。

用户可以在模块对话框中设置积分的初始状态数值，或者引用外部的信号作为初始状态。如果将积分的初始状态作为模块的参数，则可以在对话框“Initial condition source”下拉菜单中选择“internal”，然后在“Initial condition”选框中输入相应的初始数值；如果从外部信号中定义积分的初始数值，可以在“Initial condition source”下拉菜单中选择“external”，如图8.174所示。

在完成上面的属性设置以后，单击“Apply”按钮，积分器模块自动添加一个初始状态的端口，如图8.175所示。

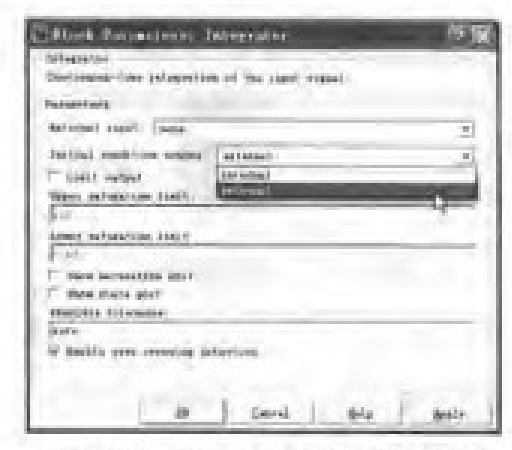


图 8.174 设置外部信号的初始状态

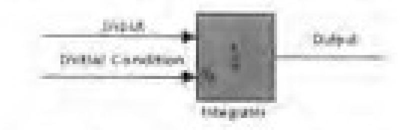


图 8.175 添加初始状态端口

8.6.5 设置积分限制

为了限制积分结果超过一定的数值，可以在对话框中选择选项，在参数选项框中输入相应的限制数值，设置这些限制数值后模块的函数变为有限积分。当积分的结果达到用户设置的限制数值后，积分器会自动停止相应的积分运算。在积分运算过程中，用户动态修改积分限制的数值，但是不能修改积分器是否限制的属性。

根据设置的不同积分限制，分下面几种情况来分析：

- 当积分器的运算结果小于或者等于最低饱和极限并且输入信号为负数的时候，输出信号则会被保持为下限数值；
- 当积分运算结果处在上限和下限之间，则直接输出积分结果；
- 当积分运算结果超过或者等于上限数值并且输入信号为正数的时候，输出信号则会被保持上限数值。

为了创建信号对象来显示积分器设定积分限制，可以在属性对话框中选中 "Show saturation port" 选项，如图 8.176 所示。

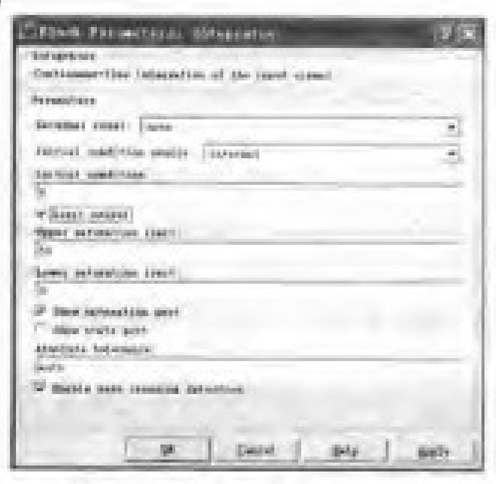


图 8.176 设置显示信号端口

选择的属性设置以后，单击对话框中的"Apply"按钮后，得到的模块如图8.177所示。

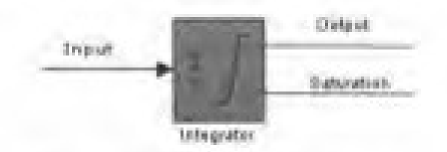


图8.177 添加信号端口

在上面步骤中添加的"Saturation"端口可以输出三种信号结果：

- 1：表示在积分器中设置了上限；
- 0：表示积分器没有设置任何的积分限制；
- –1：表示在积分器中已经设置了下限。

为了演示上面的结果，可以添加简单的模块，如图8.178所示。将"积分器"模块的属性设置为如图8.176所示，也就是将积分上限设置为50，下限设置为0，然后将仿真时间设置为100。根据基础的积分知识，积分结果为50，由于积分上限起了作用，因此"Saturation"信号端口的结果是1。

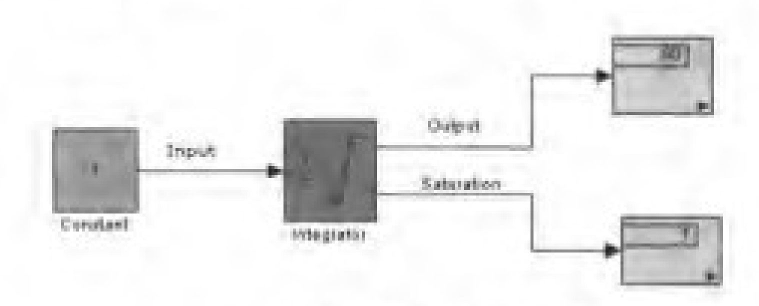


图8.178 添加简单的系统模块



在上面的积分器模块中，同时设置了积分上限和下限，但是"Saturation"信号端口的结果是1，表示在此积分器中积分上限起作用。

8.6.6 重设积分状态

在积分器模块中，可以利用外部信号信息来重新设置积分的初始状态。可以选择Simulink提供的4种重设属性中的任何一种，积分器将会启动积分重设功能。同时，在积分器模块的输入端口下面出现"触发"端口，模块中出现用户设置的重设类型。例如，可以将积分器模块设置"rasing"重设属性，如图8.179所示。

当选择属性之后，单击对话框中的"OK"按钮，得到的新模块如图8.180所示。

在积分器模块中，重设端口具有直接的数据传输通道。如果该模块的输出端口直接或者间接通过其他模块和该端口相连，则会在系统中产生代数环。为了解决这种代数环的问题，积分器模块的状态端口可以从模块的积分输出端口获取数据，而不会产生代数环。



代数环是Simulink建模过程中比较常见的，也是直接影响Simulink模型性能的问题。在后面的章节中将详细介绍关于代数环的问题。



图 8.179 设置积分器的重设属性

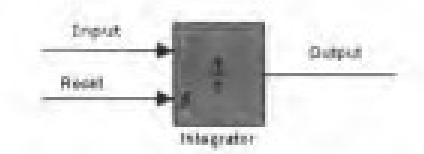


图 8.180 设置后的积分模块

8.6.7 设置积分状态端口

为了各种积分功能，Simulink允许用户设置并显示积分器模块的积分状态端口。积分器模块的积分状态端口的输出变量和模块标准输出端口的输出变量在一般情况下是相同的。当积分器模块在当前时间步长中被重设，那么积分状态端口将会显示没有重设积分状态之前的积分结果。另外，积分状态端口所输出的积分结果比输出端口所输出的结果在时间步上要早，因此就避免了代数环的问题。

可以在模块对话框中选择"Show state port"选项，来显示积分器模块中的积分状态端口，如图 8.181 所示。

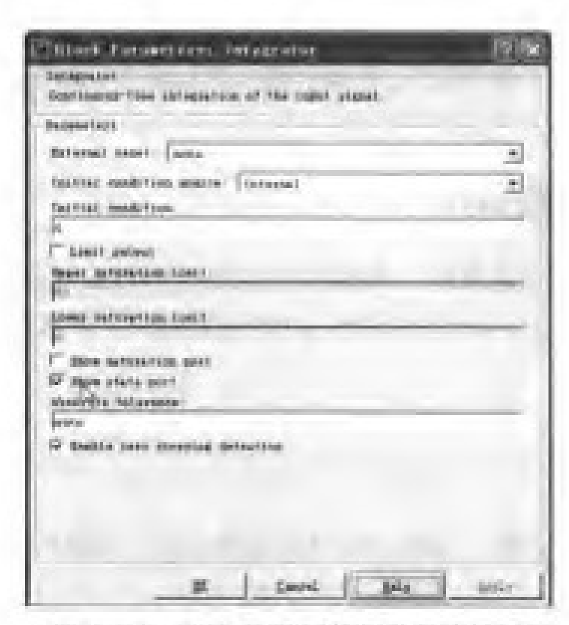


图 8.181 设置积分器的积分状态端口

当选择了属性之后，单击对话框中的"OK"按钮，得到的新模块如图 8.182 所示。

在 Simulink 中，一般有两种常见的情况需要使用到状态端口：

- 自我重设系统（见例 8.17）；
- 从一个激活的子系统向另一个子系统传递积分状态（具体的案例在后面的章节中详细介绍）。



在 Simulink 中，"状态"端口主要应用在上面提到的两种情况下，当用户更新 Simulink 的系统状态时，Simulink 会检查是否属于上面两种情况，如果不属于这些情况，Simulink 会提示系统错误。

在本小节的前面内容中已经介绍了关于积分器模块的各种常见属性端口，当选中积分所有的常见端口后，得到的结果如图 8.183 所示。

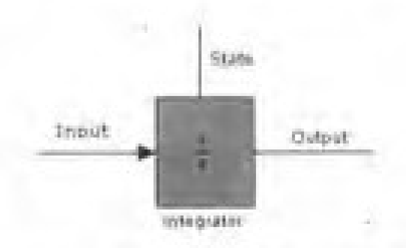


图 8.182　添加积分状态端口后的模块

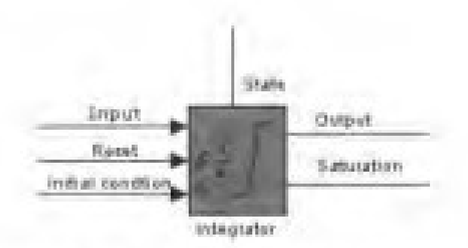


图 8.183　选中所有的积分器属性



在 Simulink 中，"Integrator"模块可以在其数据端口中接收或者输出双精度的信号数据，而在其外部重设端口则可以接收双精度或者布尔数据类型。

8.7　非线性系统建模

尽管在前面列举了很多关于线性系统的实际案例，但是在实际情况中严格意义上的线性系统是很少的，大量的系统或者原件是非线性的。为了完成这些内容的仿真系统，Simulink 在模块库中提供许多典型的非线性模块。相对于线性模块，非线性模块更加复杂，本节还将使用简单的案例来介绍非线性模块的使用方法。

8.7.1　非线性系统建模简介

例 8.22　使用 Simulink 创建系统，求解非线性微分方程$(3x-2x^2)x'-4x=4x''$。其中 x 和 x' 都是时间的函数，也就是 $x(t)$ 和 $x'(t)$，其初始值为，$x'(0)=0,x(0)=2$。求解该方程的数值解，并绘制函数的波形。

step 1　改写上面的微分方程，将上面的方程改写为下面的等式

$$\frac{1}{4}(3x-2x^2)x'-x=x''$$

step 2　根据上面改写的求解方程，创建 Simulink 仿真系统，如图 8.184 所示。

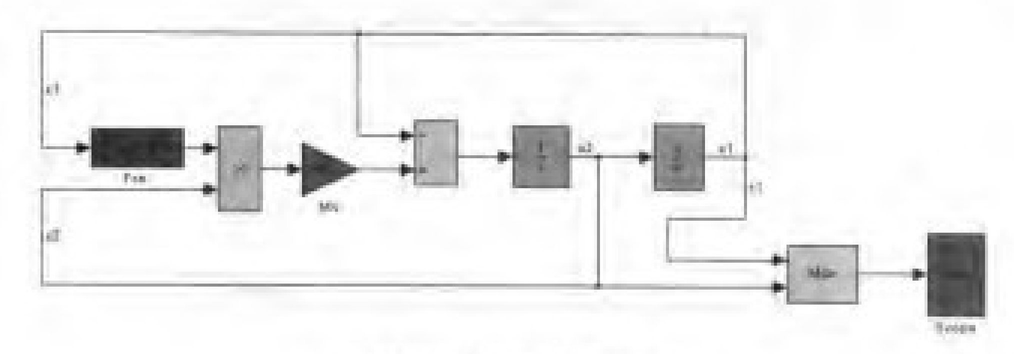


图 8.184　创建的系统模块

在模型中，使用了很多新的系统模块，例如 Fcn、Product 等，这些模块将在非线性微分方程中经常用到，在后面的步骤中将详细介绍这些模块的属性设置。



在以上步骤所创建的系统模块中，将 x' 设置为 x2，将 x 设置为 x1，之所以这样设置，是为了加强系统模块的可读性。

step 3 设置 "Fcn" 模块的属性。双击上面系统中的 "Fcn" 模块，打开对应的属性对话框，在其中设置表达式，如图 8.185 所示。

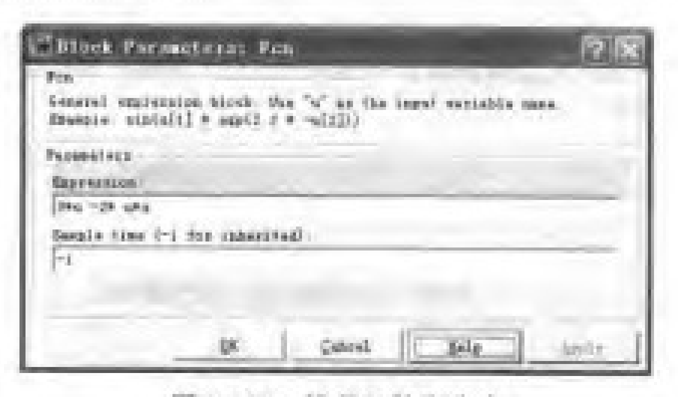


图 8.185　设置函数表达式

在对话框中的 "Expression" 选框中输入 "3*u -2*u*u"，其中 u 代表输入该模块信号的变量，在本实例中也就是信号变量 x。在 Simulink 中，Fcn 模块支持所有 C 语言条件下的所有相关表达式。在该表达式中可以包含变量 u、数值常数、数学运算符、关系运算符、逻辑运算符、圆括号、数学函数和 MATLAB 工作空间中的变量等。关于该模块的其他信息，请查看相关的帮助文件。



在 Simulink 中，"Fcn" 模块不支持可变的参数，在系统仿真过程中该模块忽略所有工作空间中变量的数值变化，同时，"Fcn" 模块也不支持 Custom Storage 数据类。

step 4 设置 "Product" 模块的属性。双击上面系统中的 "Product" 模块，打开对应的属性对话框，在其中设置表达式，如图 8.186 所示。

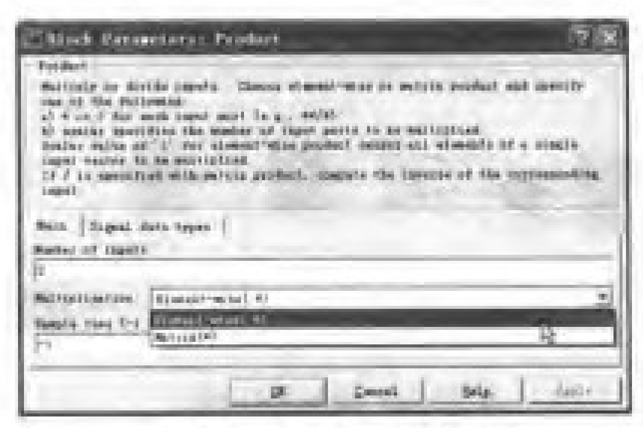


图 8.186　设置乘法模块的属性

在上面的对话框中，选择 "Product" 模块的 "Multiplication" 属性为 "Element-wise(.*)"，表示对模块输入变量进行点乘运算，在 "Number of signals" 选框中输入信号的个数为 "2"，这样 "Product" 模块会提供 2 个输入端口。



在 Simulink 中，"Product" 模块对输入变量的维度和个数有着匹配的要求，同时也可以选择不同的相乘模式，当选择该模块时需要根据系统的要求选择不同的属性。

step 5 设置 "Integrator" 模块的属性。在本实例中使用的积分器模块使用的都是默认属性端口，用户主要需要为其设置初始数值条件，以模块 "x1" 为例，双击该模块，打开属性对话框，设置的初始数值如图 8.187 所示。



图 8.187　设置积分器模块的属性



对于上面系统中的其他模块属性，这里就不详细介绍了，感兴趣的读者可以查看前面章节中的相关内容。

step 6 查看系统的仿真图形结果。将系统仿真时间设置为 20，然后运行仿真，双击上面系统模块中的 "Scope" 模块，查看仿真结果，如图 8.188 所示。

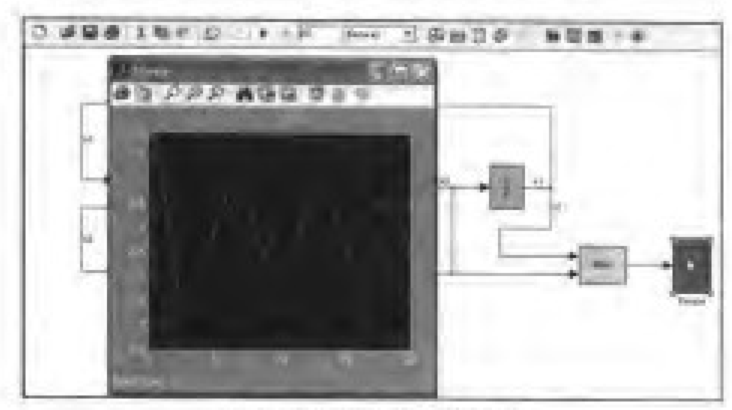


图 8.188　查看仿真结果

在上面的仿真结果中，黄色的曲线表示变量 $x(t)$，红色的曲线表示 $x'(t)$。

step 7 修改仿真的模块。为了能够在 MATLAB 的工作空间中演示上面的仿真结果，添加新的系统模块，如图 8.189 所示。

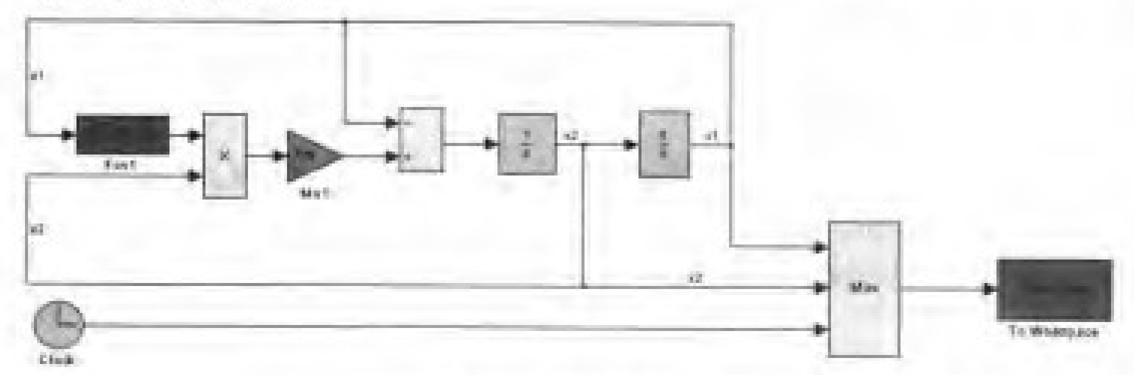


图 8.189　添加新的模块

step 8 设置"Mux"模块的属性。为了在 MATLAB 的工作空间中显示时间变量，将"Mux"模块的输入端口改为 3，如图 8.190 所示。

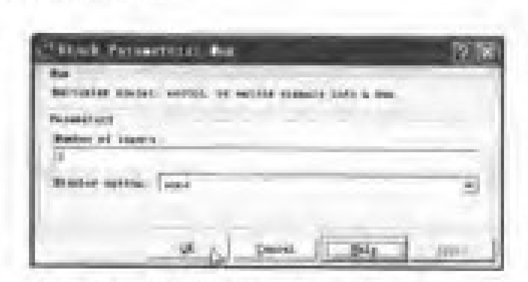


图 8.190　修改"Mux"模块的输入端口

step 9 将系统的仿真时间修改为 30，运行系统仿真，得出输出变量"SimuData"，然后返回 MATLAB 的命令窗口中输入下面的代码：

```
>> x=SimuData(:,1);
>>dx=SimuData(:,2);
>> t=SimuData(:,3);
>> plot(t,x,'b',t,dx,'y','LineWidth',2),hold on
>> grid
>> xlabel('时间(t)')
>> legend('x','dx')
```

step 10 查看图形结果。在输入代码后，按“Enter”键，得到的图形如图 8.191 所示。

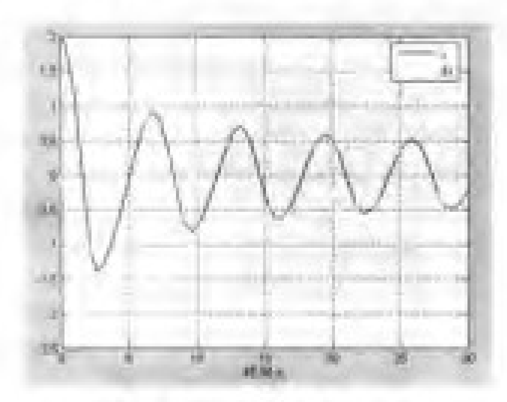


图 8.191　绘制图形结果



从上面的实例中可以看出，使用“Scope”模块绘制的函数图形和 MATLAB 的典型函数绘制的结果相同，表明程序模块设计正确。

8.7.2　非线性系统建模实例

在本小节中，将以具体的实例，演示如何在 Simulink 中创建非线性系统。

例 8.23　使用 Simulink 创建系统，求解一个非线性摩擦模型。根据基础的物理模型，假设物体的位移为 x，而且该物体的摩擦力微分方程如下：

$$\frac{1}{\delta}(kx - F_fr) = x''$$

在上面的微分方程中，k、δ 都是方程的系统参数，F_fr 表示物体的摩擦力是物体运动速度的非线性函数。也就是说，$F_fr=F_fr(x')$。再根据基础的物理知识得知，物体的摩擦力包括静态摩擦力和动态摩擦力两种。根据物体运动的不同速度，物体运动所受的摩擦力类型会有不同，物体会在不同的速度范围内接受不同的摩擦力类型。

本例分析上面模型中的一种情况，其中初始位置 x_0=−0.5，初始速度 x_0'=100。而且物体的位移范围为 $-3.5 \leqslant x \leqslant 5.5$。而且静态摩擦力 f_s=80，动态摩擦力 f_k=40，同时，为了分析的方便，将方程中的参数设置为 δ=1，k=−100。

step 1 根据上面的微分方程，添加相应的系统模块，如图 8.192 所示。

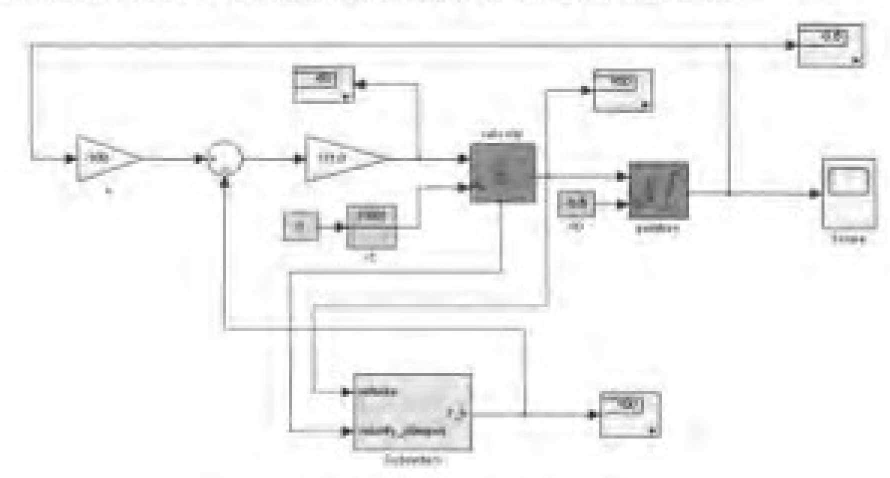


图 8.192　添加微分方程的模块

在上面的程序模块中，包括两个主线，其中第一条主线的系统模块代表的是前面包括的微分方程$\frac{1}{\delta}(kx-F_fr)=x''$，同时示波器中显示物体位移的波形。在本步骤中，为了简化分析，同时由于$F_fr=F_fr(x')$，为计算F_fr的数值，添加了子系统模块。该子系统模块中的输入变量包括velocity和velocity_stateport，其中变量velocity_stateport是为了避免在后面步骤中出现代数环，是由“velocity”模块的“状态”端口得到的输出变量。

step 2 设置“velocity”模块的属性。双击“velocity”模块，打开模块的属性对话框，让其显示“状态”端口，如图8.193所示。



图 8.193　设置积分器模块的属性



在上面的系统模块中，添加了多个“Display”模块，这样做似乎让整个系统模块显得十分臃肿，但是，有一个明显的好处在于，可以让用户在仿真过程中查看各个变量的数值，有利于分析整个仿真系统。

step 3 设置计算摩擦力的子系统模块。双击上面的子系统模块，在打开的模块编辑器中添加子系统模块，如图8.194所示。

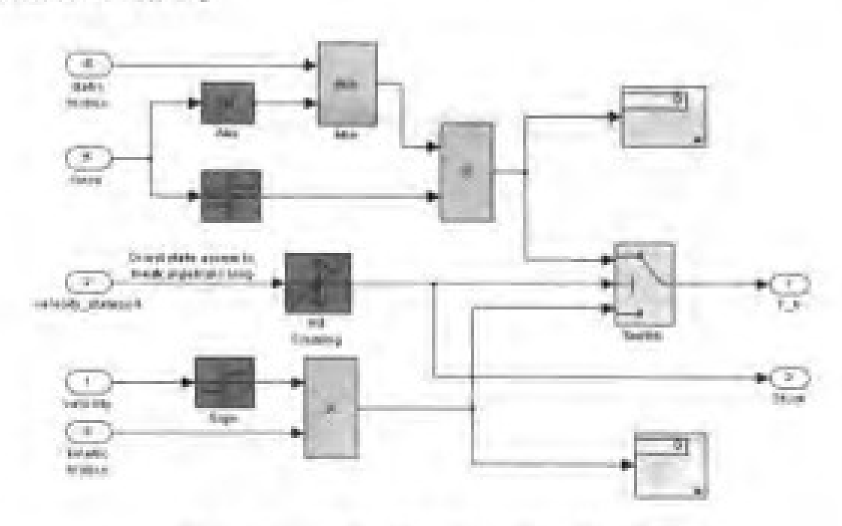


图 8.194　添加计算摩擦力的子系统模块

step 4 查看程序模块的变化。当添加程序模块后，原来的程序模块会随着变化，得到的结果如图8.195所示。

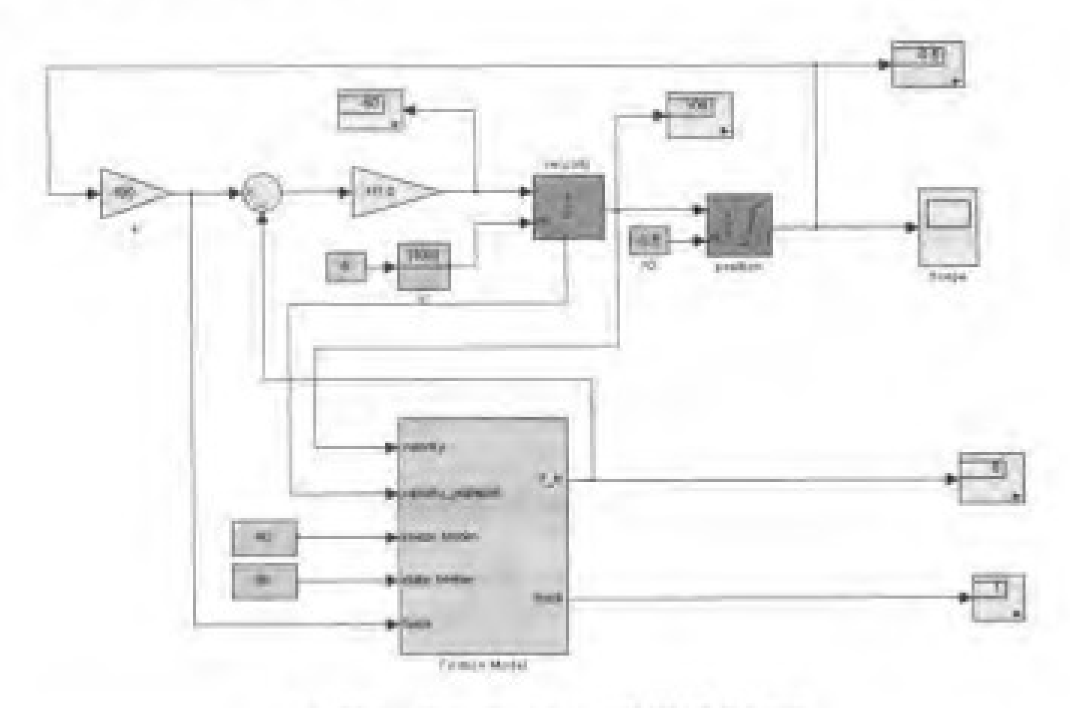


图 8.195　添加子系统模块后的主系统模块

和原来的系统模块相比，子系统模块多了三个输入端口，分别表示静态摩擦力、动态摩擦力和外部推力等，分别在该子系统模块中添加新的输入变量，下面将详细分析添加的子系统模块。



尽管在本步骤中涉及子系统的概念，但是在这里笔者并不打算详细介绍该子系统的内容。因为子系统（subsystem）在Simulink中是十分重要的内容，将在后面的章节中专门介绍，在这里可以将子系统当作一个黑箱，输入所有和摩擦力相关的变量，输出的是物体运动的摩擦力。

step 5 设置“switch”模块的属性。在上面的子系统模块中，“switch”模块是子系统的“选择”开关，该模块根据物体运动的速度和外力情况来得出物体运动的摩擦力数值。其对应的模块属性对话框如图 8.196 所示。

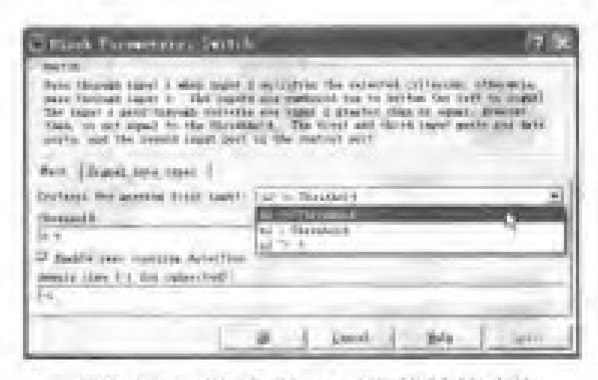


图 8.196　设置“Switch”模块的功能

step 6 查看上部分信号。下面将根据该“Switch”模块的各条信号，分析摩擦力的数值情况。其中，和该模块顶部信号端口相连的信号如图 8.197 所示。

该部分信号系统的目的在于分析物体的静摩擦力数值，首先从4号输入模块输入物体最大静摩擦力的数值（static friction），从 5 号输入模块中输入物体所受的外部推力。根据物理

知识得知，当外部推力小于最大静摩擦力时，物体所受的摩擦力数值就是外部推力；当外部推力大于最大静摩擦力一定的数值后，物体才会开始运动。因此，添加 "Abs" 模块将求得外部推力的数值，然后使用 "Min" 模块来得最大静摩擦力和外部推力的最小值，得到物体所受的静摩擦力数值，用 "Display" 模块显示该数值。

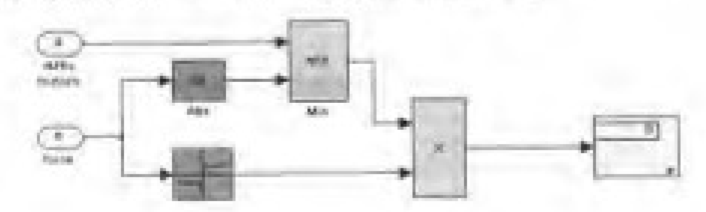


图 8.197 子系统的上部信号系统



在上面的模块中添加了 "singum" 模块，该模块的主要功能在于确定摩擦力的数值符号，最后得到的摩擦力数值在 "Display" 模块中显示。

step 7 查看下部分信号系统。下面分析该子系统中的下部分信号系统，如图 8.198 所示。

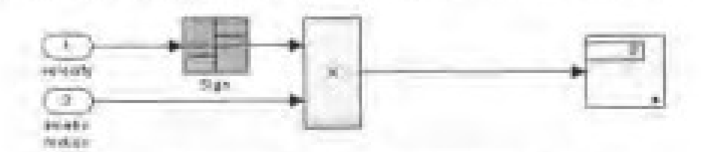


图 8.198 子系统的下部信号系统

该部分系统模块处理的主要是物体运动时的摩擦力数值，物体运动的速度和动摩擦力系数的乘积就是物体所承受的动摩擦力数值。

step 8 设置两个积分器模块的控制模块。这些系统模块的功能是设置整个系统的积分属性，添加后的模块系统如图 8.199 所示。

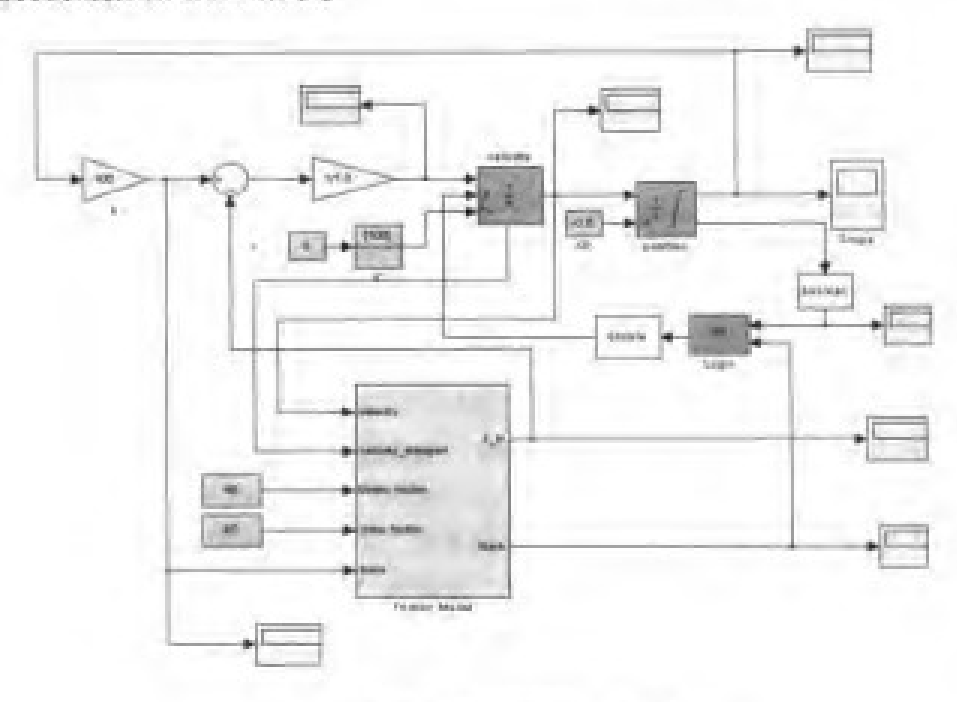


图 8.199 设计完成的系统模块

step 9 设置 "Velocity" 积分器模块的属性。双击 "Velocity" 积分器模块，打开对应的属性对话框，如图 8.200 所示。

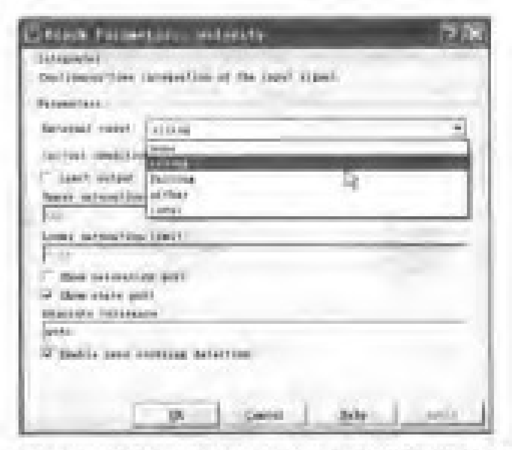


图 8.200　设置 "Velocity" 积分器模块的属性

将 "Velocity" 积分器模块的重设属性设置为 "rising"，具体的重设属性将在后面的步骤中详细设置对应的逻辑模块实现。

step 10 设置 "Position" 积分器模块的属性。双击 "Position" 积分器模块，打开对应的属性对话框，在其中设置其属性，如图 8.201 所示。



图 8.201　设置 "Position" 积分器模块的属性

选中 "Show saturation port" 选项，积分器模块显示 "Saturation Port"。由于在前面的步骤中，设置该位移变量积分的上下限，因此添加该端口可以显示积分达到限制的类型，该端口输出的数据类型是布尔数据类型。

step 11 设置 "Data Type Conversion" 模块的属性。由于积分器模块的所有输入变量数据类型都是 "double"，因此在上面的系统模块中添加了 "Data Type Conversion" 模块，将数据类型在 "Boolean" 和 "Double" 中实现转换，选择数据转换模块的属性，如图 8.202 所示。

在上面的对话框中，在 "Output data type mode" 选择 "double" 数据类型，表示该模块会将输入模块的数据转换为 "double" 数据类型。对于该系统模块中的另外一个数据转换模块，可以将其 "Output data type mode" 设置为 "Boolean" 数据类型就可以了。

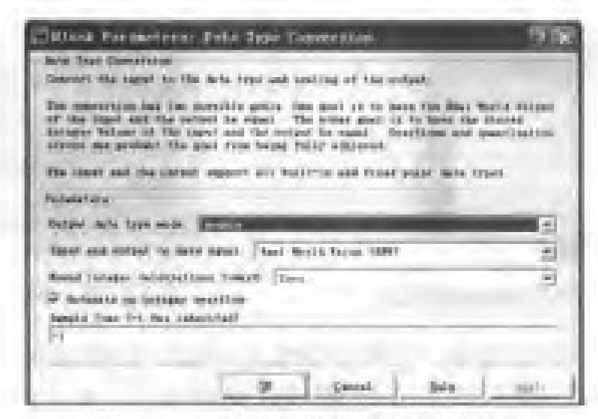


图 8.202 设置数据转换模块的属性



"Data Type Conversion" 模块在Simulink建模中有着广泛的应用，可以实现多种数据类型之间的转换，还可以起到数据保存方式之间的转换等多种功能，灵活运用该模块可以实现多种功能，关于该模块的详细用法将在后面章节中详细介绍。

step 12 设置仿真时间，运行整个仿真系统，得到仿真结果。将系统的仿真时间设置为 2，然后运行仿真，得到的结果如图 8.203 所示。

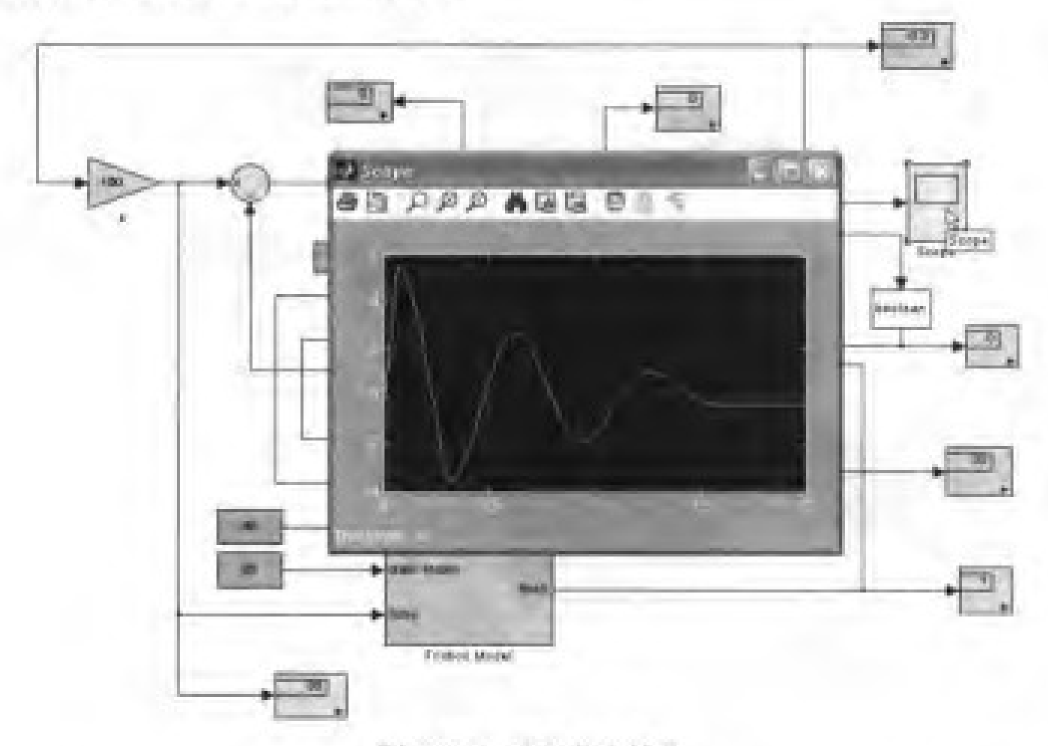


图 8.203 查看仿真结果

在上面的仿真结果中，各个显示模块中显示在 t=2s时，物体的运动速度、位移、摩擦力等各个物理参量的数值，同时在 "Scope" 模块中显示物体位移随时间变化的曲线。

step 13 修改坐标轴数值范围。为了更好地显示物体移动的变化情况，可以修改图形中Y轴坐标数值的显示范围，如图 8.204 所示。

step 14 查看修改后的图形。单击对话框中的 "Apply" 按钮，查看修改后的图形情况，如图 8.205所示。


图 8.204　修改图形的 Y 轴坐标数值


图 8.205　修改后的显示图形

从上面的仿真结果中可以看出，在初始位置 x_0=−0.5，初始速度 x_0'=100，同时在外部摩擦力系数的条件下，物体会停止在 x=−0.3 的位置，最终停止，并且开始静止的时间约为 t=1.6s 左右。因此，从上面整个仿真系统的条件下，物体从 x_0=−0.5 的位置，以初速度 100 的条件下开始运动，在变动的外力条件下，最终在 t=1.6s 的时刻，停止在 x=−0.3 的位置，最终保持静止。

step 15 修改系统仿真的参数，得到新的仿真结果，如图 8.206 所示。

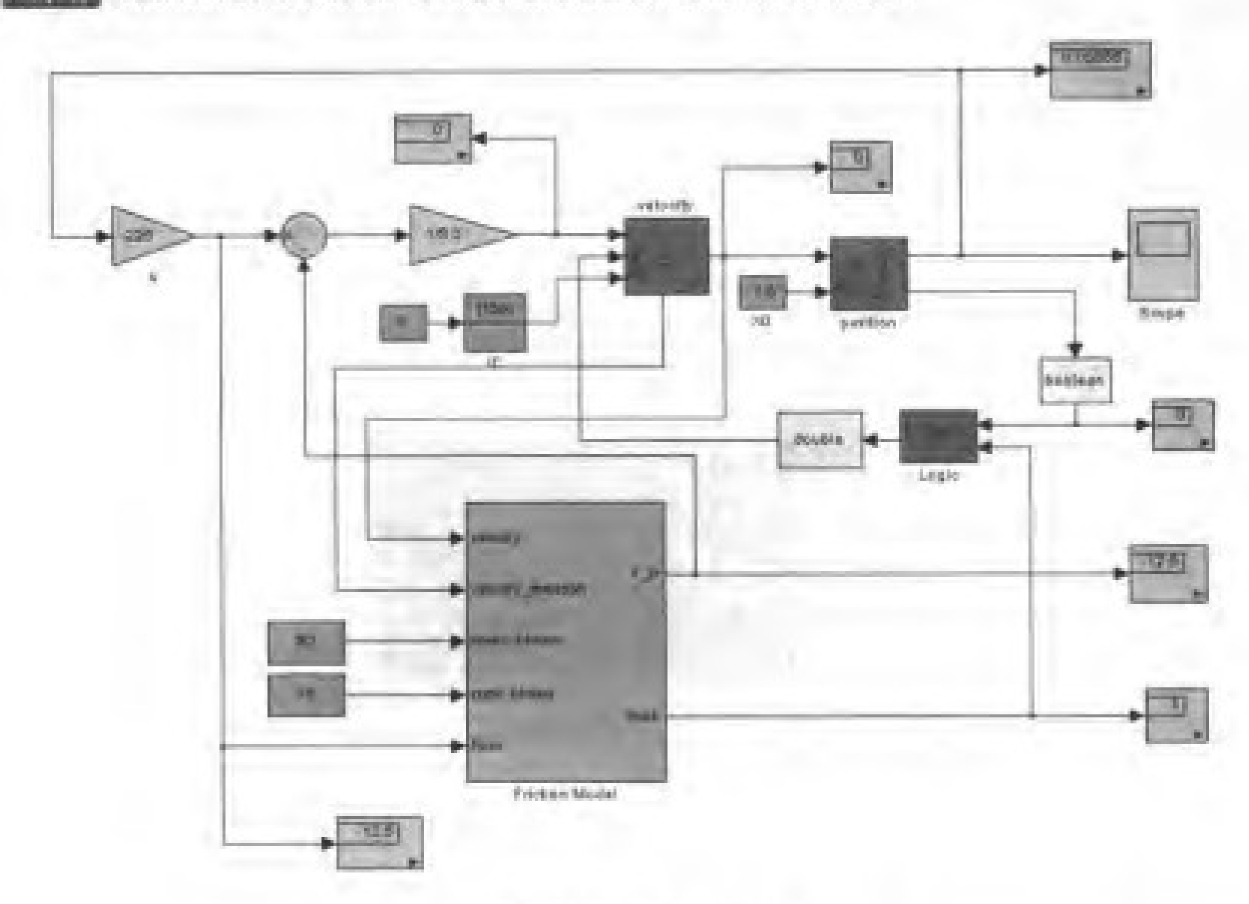


图 8.206　修改仿真参数

在上面的仿真中，设置初始位置 x_0=−1.5，初始速度 x_0'=150，静态摩擦力 f_s=75，动态摩擦力 f_k=50，微分方程系数 δ=3.5，k=−225。

step 16 运行仿真。将仿真时间设置为 4，进行仿真，得到的结果如图 8.207 所示。

从上面的仿真结果可以看出，当修改物体运动的环境后，最终物体会在 t=3.6s 的时刻，停止在 x=0.06 的位置上，最终保持静止。

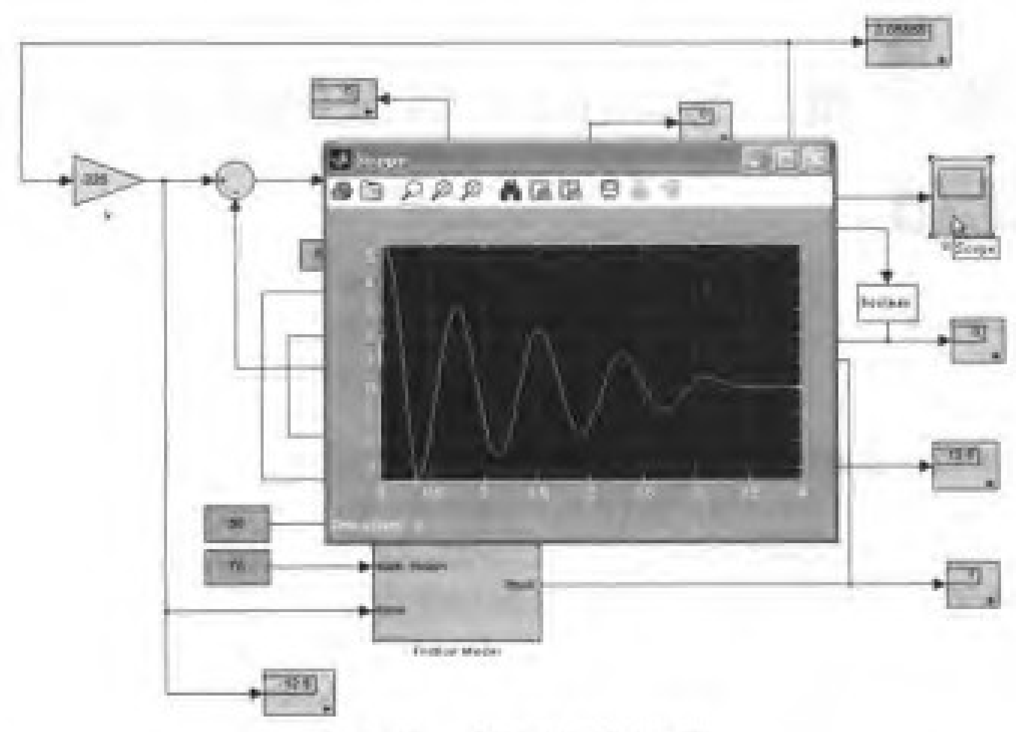


图 8.207 重新运行系统的仿真

8.8 小结

本章主要向读者介绍了 Simulink 仿真系统的基础知识，主要包括了 Simulink 的数据类型、基础操作、信号类型、仿真属性设置等，在最后两节介绍了使用 Simulink 实现线性系统建模和非线性系统建模。本章内容是 Simulink 仿真的基础章节，希望通过本章内容，加深读者对 Simulink 的理解。在后面的章节中，将主要介绍 Simulink 的高级技术。

◆ ◆ ◆

第9章　Simulink仿真的高级技术

本章包括

- 子系统
- 使能子系统
- S函数
- 封装子系统
- 触发子系统
- 仿真结果分析

在上一章中已经详细分析了关于Simulink的一些基础知识。实质上，Simulink在电力、航空、自动化等各个领域中有着广泛的应用，完成各种十分复杂的功能。为了完成这些功能，需要了解关于Simulink的一些重要的高级技术，例如，子系统、封装、S函数等。同时，Simulink作为MATLAB中的一个重要组件，也可以和MATLAB的其他组件结合起来使用，例如，Simulink和GUI、Virtual Reality Toolbox等重要组件配合起来使用，完成比较复杂和综合的功能。

在本章中，将按照循序渐进、由浅入深的次序介绍关于Simulink的各种高级技术。首先，将使用各种例子来介绍Simulink子系统、封装的概念，然后，介绍关于Simulink的S函数的作用等。最后，作为介绍Simulink的最后章节，还将介绍Simulink性能改进、调试等各方面的内容。

9.1　子系统

上一章已经接触到一些关于子系统的实例，对Simulink的子系统有了直观的印象。本节将使用几个比较典型的实例，详细介绍如何使用Simulink来创建子系统，以及关于子系统的各种注意事项。

9.1.1　子系统的基础知识

当使用Simulink来解决或者进行仿真比较综合的实际问题时，模型结构会变得比较复杂，涉及的模型原件也会变得繁多。如果所有的原件都是在原来的模块基础上添加的，那么模型的阅读会变得越来越艰难。这个时候，如果使用子系统则可以解决这个主要矛盾。这样，将复杂的模型分割为几个比较小的模块，可以使得整个模块更加简洁、可读性更高，也可以方便用户操作比较复杂的模型。

子系统除了可以解决上面的问题之外，还具有以下几个重要的优点：

- 减少模块窗口中模块的个数，使得模型窗口更加简洁；
- 将一些功能相关的模块集成在一起，而且可以实现复用（Reuse）的功能；
- 提高整个系统运行的效率和可靠性；
- 符合面向对象的概念，方便用户分析研究系统时进行概念抽象。



在仿真建模中的子系统的作用，类似于在MATLAB中的M函数文件，C语言中的Function sub-programs，FORTRAN语言中的subroutine sub-programs等。

在Simulink中，可以使用如下两种方法创建子系统。

- 通过子系统模块来创建子系统：也就是先向模型中添加"Subsystem"模块，然后打开该模

块并向其中添加模块。
- 结合已经在模块编辑器中添加的模块，创建子系统。



在Simulink中，之所以提供两种创建子系统的方法，是因为这两种方法都有各自的适用范围，这些内容将在后面详细介绍。

9.1.2 使用子系统模块创建子系统

例 9.1 使用Simulink中的Subsystem模块来创建系统，系统的输入变量为 N，输出的变量为从1到 N 的自然数的累积求和数值，下面分步骤详细介绍。

step 1 单击"Simulink Library Browser"对话框中的按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "Model"命令，打开一个空白模型窗口，然后添加系统模块，如图9.1所示。

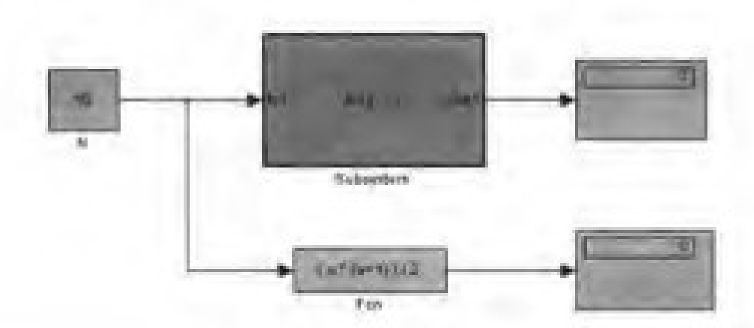


图 9.1 添加程序模块

在上面模块中的"Subsystem"子模块属于"Ports & Subsystems"模块库中的"For Iterator Subsystem"模块，该模块完成的主要功能相当于C语言中的For循环。由于本实例比较简单，本例选用的是该模块的默认属性，如图9.2所示。



图 9.2 Subsystem 的属性对话框

step 2 设置"Fcn"模块的属性。在上面的程序模块中，为了检验子系统计算的结果，在下面的分支中添加了"Fcn"模块，属于"User-Defined Functions"模块库下的"Fcn"模块，其模块属性如图9.3所示。

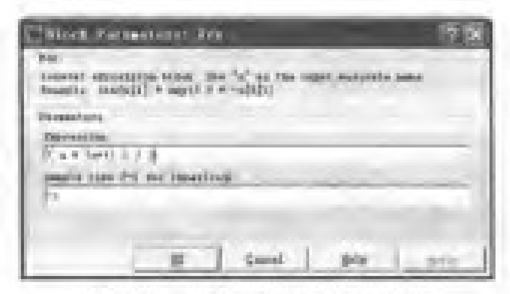


图 9.3　"Fcn" 模块的属性

根据基础的数学知识，自然数从 1 到 N 的叠加求和公式为：

$$1+2+3+\cdots+N=\frac{N(N+1)}{2}$$

因此，在上面的 "Fcn" 模块中，就是通过该公式得到的求和数值结果。

step 3 双击 "Subsystem" 模块，打开模块编辑窗口，添加该系统的模块，如图 9.4 所示。

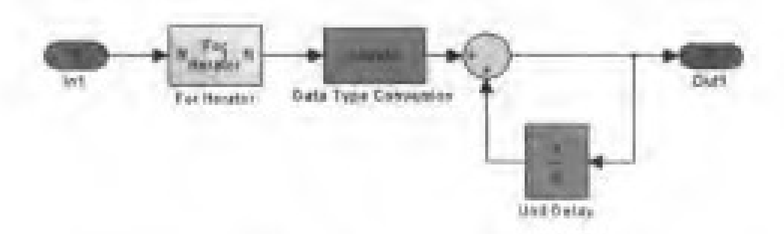


图 9.4　编写子系统模块

step 4 设置 "For Iterator" 模块的属性。对话框在上面的子系统模块中添加了 "For Iterator" 模块，该模块的功能在于控制 For 的循环条件。该模块的属性对话框如图 9.5 所示。

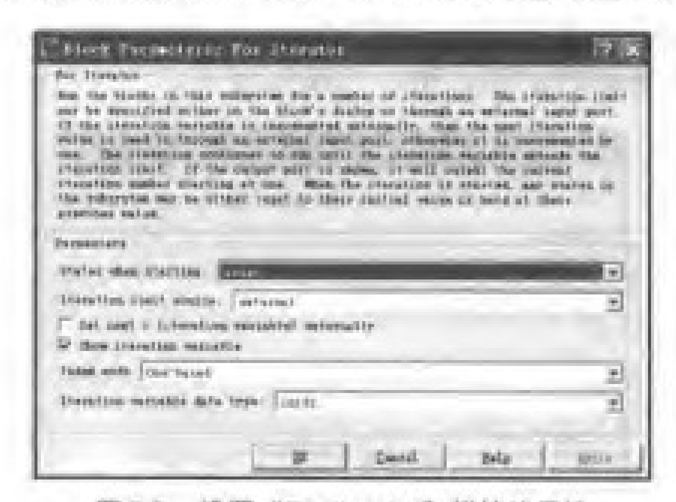


图 9.5　设置 "For Iterator" 模块的属性

在该对话框中，设置 "For Iterator" 模块的循环属性，循环的输出数据为 "int32"，然后使用 "Data Type Conversion" 模块，将该输出数据转换为 "double" 数据类型。



关于上面系统中的 "For Iterator" 模块和 "Data Type Conversion" 模块的具体功能，以及对应的属性设置功能，将在后面详细介绍。

step 5 保存子系统模块，然后运行仿真结果，得到的结果如图 9.6 所示。

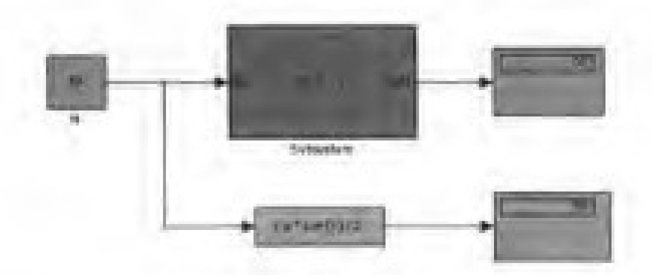


图 9.6 查看系统仿真的结果

当输入计算自然数的数值是10时，仿真系统得出结果55。可以重新输入叠加自然数的数值，例如20，重新运行仿真，得到的结果如图9.7所示。

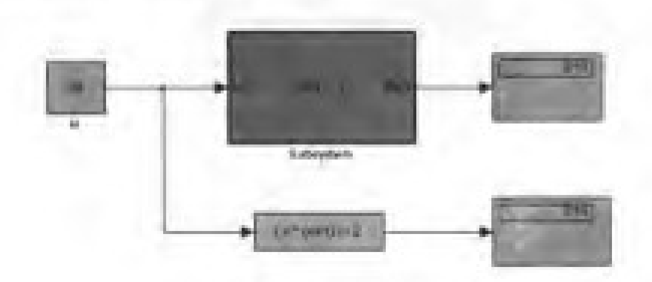


图 9.7 重新运行仿真

从上面的结果中可以看出，当输入计算的自然数为20时，仿真结果为210。上面的实例比较简单，但是演示了创建子系统的一般步骤，具体的内容为：

step 1 从模块库中选择对应的子系统模块，添加到模块编辑器中；

step 2 双击相应的子系统模块，打开子系统的模块编辑器；

step 3 在模块编辑器中添加模块，创建子系统，然后保存；

step 4 运行仿真，得到结果。



说明：在Simulink中，提供多个复杂的子系统模块，不同的模块具有不同的属性设置情况，这些内容将在后面详细介绍。

9.1.3 使用模块组合子系统

例 9.2 使用Simulink编写系统模块，求解微分方程$\sin x - \frac{1}{2}x = x t$的数值解。

step 1 单击"Simulink Library Browser"对话框中的□按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "Model" 命令，打开一个空白模型窗口，然后添加系统模块，如图9.8所示。

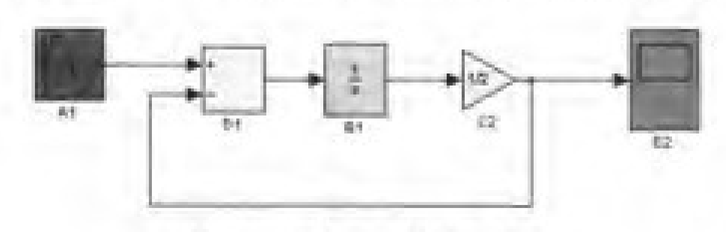


图 9.8 添加相应的系统模块



上面的程序模块比较简单，属于Simulink的线性系统问题，具体的创建方法可以参看前面章节中的相关内容。

step 2 运行上面的仿真系统，得到的结果如图9.9所示。

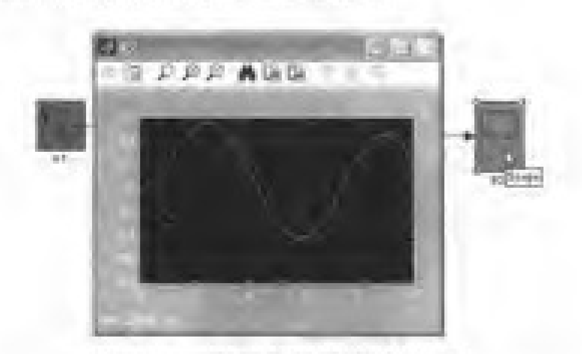


图9.9 查看系统的仿真结果

step 3 选中模块D1、B1，然后选择"Edit"→"Create Subsystem"命令，将选中的模块编辑为子系统，如图9.10所示。

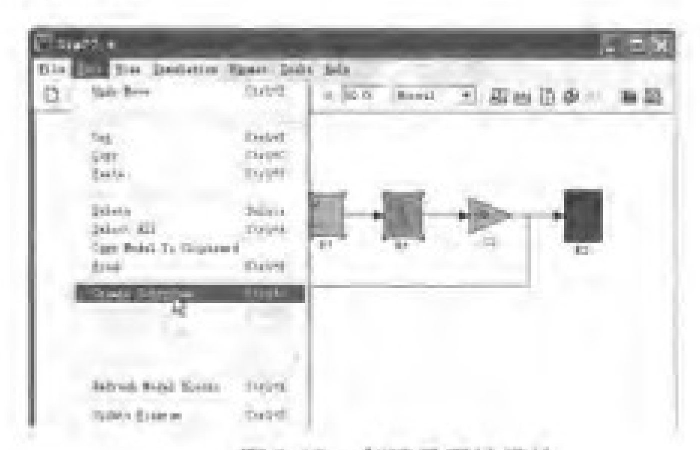


图9.10 创建子系统模块

step 4 查看创建子系统后的模块。选择上面菜单选项后，Simulink就会将选中的模块设置为"Subsystem"模块，适当编辑模块的位置和颜色，结果如图9.11所示。

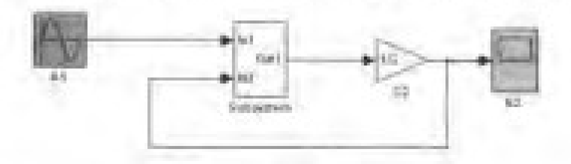


图9.11 创建子系统

step 5 查看子系统模块。双击上面的子系统模块，打开对应的子系统模块编辑器，查看Simulink自动创建的子系统模块，如图9.12所示。

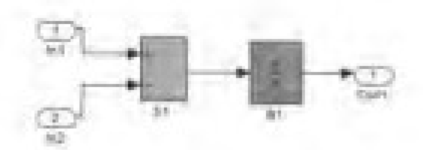


图9.12 Simulink创建的子系统

从上面的子系统模块中可以看出，Simulink会对选用的模块添加对应的Import和Output数据模块，作为和上一级模块实现数据传递的端口。

step 6 重新进行系统仿真。当完成上面的子系统创建过程后，可以重新运行系统，进行系统的仿真，得到的仿真结果如图9.13所示。

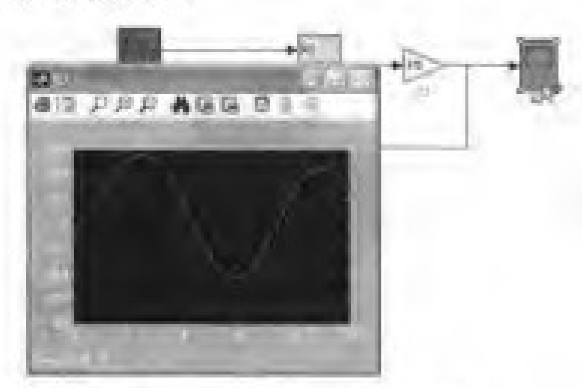


图9.13　重新运行系统的仿真

上面的实例比较简单，但是演示了如何通过套装来创建子系统的基本步骤：

step 1 按照系统的要求，创建对应的系统模块；

step 2 选择希望创建子系统的系统模块，选择"Edit"⇨"Create Subsystem"命令；

step 3 双击创建的子系统模块，编辑子系统的名称或者其他属性；

step 4 运行仿真，得出结果。



使用上面两种方法创建的子系统都是比较常见的，但是在最后的结果效果上还是有明显差别的。使用第一种方法创建的子系统在执行过程中有"优先级"的差别，而使用第二种方法创建的子系统则没有"优先级"的差别。

9.2 子系统实例

在本节中将使用一个比较综合的例子来介绍如何在Simulink中创建子系统，以及如何实现子系统之间的数据传递。

9.2.1 添加控制信号

例9.3 利用Simulink来创建仿真系统，该仿真系统中包含两种数据信号：方形波脉冲和正弦波图形。其中，正弦波是输入信号，方形波脉冲是控制信号。需要使用子系统模块来输出三种类型的信号：原始的正弦波信号，经过控制信号处理后的两种正弦波信号。具体的信号属性将在后面步骤中详细介绍。

step 1 根据该仿真系统的功能要求，需要添加对应的程序模块，如图9.14所示。

step 2 设置方形波的属性。双击上面的"Pulse Generator"模块，打开对应的属性对话框，在其中设置方形波的参数属性，如图9.15所示。

在本实例中，方形波的主要功能是控制信号的输出情况，因此，属性将直接影响信号的输出

结果。在Simulink中，"Pulse Generator"模块可以生成固定间距的方形波，关于方形波的主要参数包括Amplitude（振幅）、Pulse Width（脉冲宽度）、Period（周期）和Phase Delay（相位延迟）等。参见图9.16了解到各属性的含义。

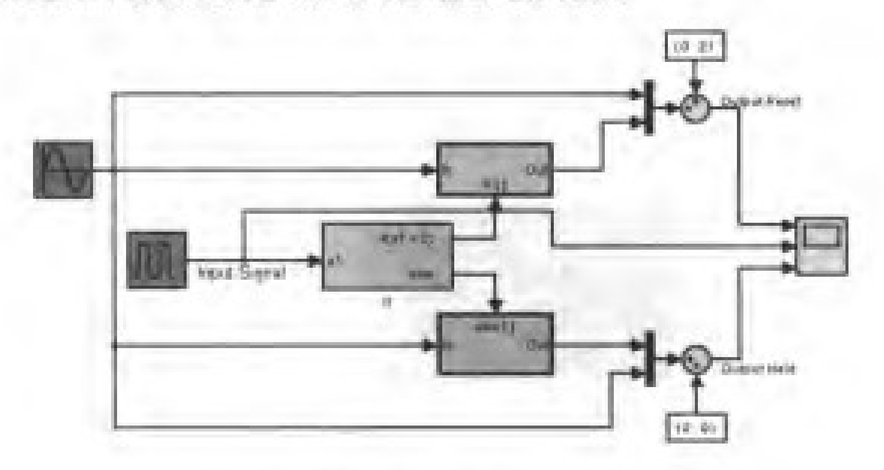


图9.14 添加程序模块



图9.15 方形波的属性

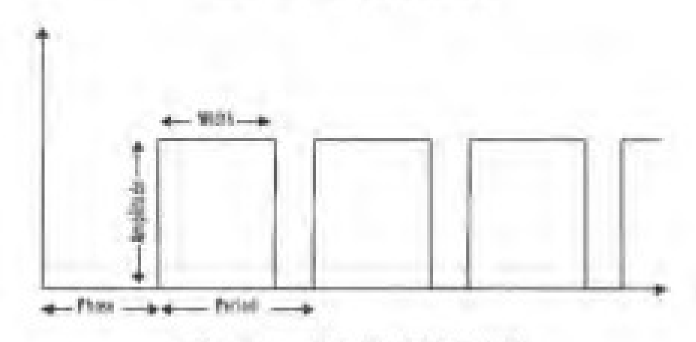


图9.16 方形波的属性参数

step 3 显示方形波的重要参数的数值。为了增加整个仿真系统的可读性，可以显示方形波的重要参数数值。选择"Pulse Generator"模块，然后右击鼠标，在弹出的快捷菜单中选择"Block Properties"选项，打开"Block Properties: Pulse Generator"对话框，选择"Block Annotation"选项卡，在其中设置显示的参数数值，如图9.17所示。

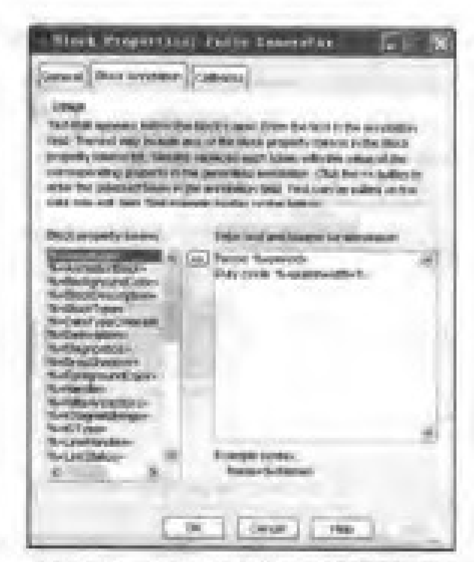


图 9.17　设置模块显示的参数数值

step 4 查看修改后的程序模块。单击上面对话框中的 "Apply" 按钮，就可以查看修改后的程序模块，如图 9.18 所示。

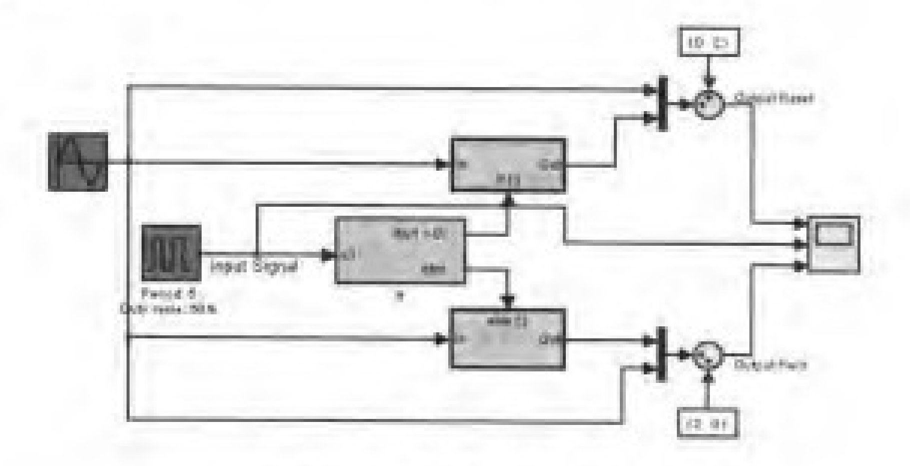


图 9.18　显示模块的主要参数



关于模块参数设置的情况，在前面已经介绍过，这里就不详细介绍了，感兴趣的读者可以查看前面章节中的相关内容。

9.2.2 添加子系统模块

延续上面小节的步骤。

step 1 添加子系统模块的说明名称。为了更方便地查看系统，需要添加子系统模块的说明名称，如图 9.19 所示。

step 2 设置 "If" 模块的属性。双击上面系统中的 "If" 模块，打开对应的模块属性对话框，设置该模块的属性，如图 9.20 所示。

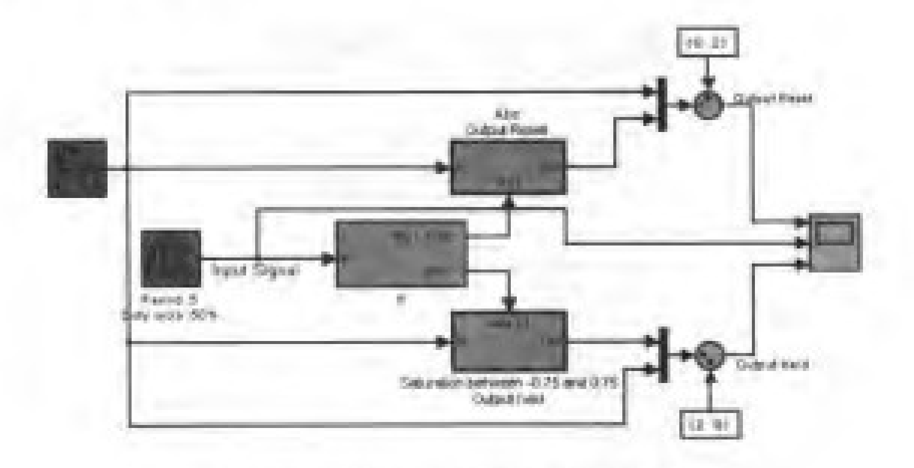


图 9.19　添加子系统模块的说明文字



图 9.20　设置"If"模块的属性

在上面的对话框中，将 IF 模块的输入变量个数设置为"1"，同时 If 表达式为"u1 > 0"，其中 u1 代表输入信号的变量。由于该实例中，If 判断表达式只有一个条件，因此在"Elseif expression"文字框中不需要输入任何的表达式。最后，由于本实例中具有两种情况下的处理模块，因此，需要选中"Show else condition"选框，其他属性采用系统的默认情况。

step 3 分析 IF 模块的功能。在 Simulink 中，"Ports & Subsystems"模块库下的 If 模块主要功能是在 Simulink 中执行类似于 C 语言中的 if-else 控制流程序语句。If 模块和若干包含执行端口的"If Action"子系统模块执行标准的 C 语言中的 if-else 逻辑功能，其典型的模块结构系统如图 9.21 所示。

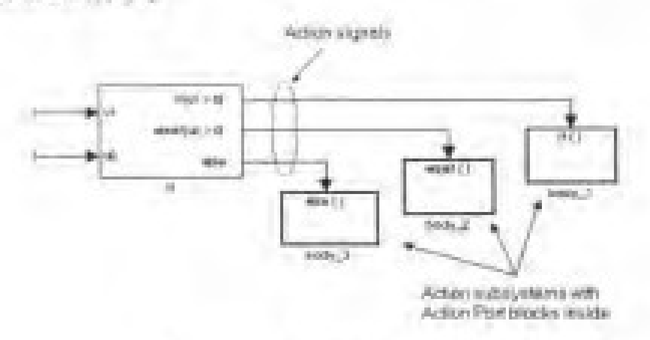


图 9.21　If 模块的典型执行结构

针对上面的系统模块结构，其功能和下面的程序代码相同：

```
if (u1 > 0) {
    body_1;
}
elseif (u2 > 0){
    body_2;
}
else {
    body_3;
}
```



和C语言类似，If模块在Simulink中可以起到各种复杂的功能，灵活使用该模块可以解决比较复杂的实际问题。这里限于篇幅，就不详细展开介绍了，感兴趣的读者请查阅相应的帮助文件。

step 4 设置“If Action”子系统模块的属性。双击“If Action”子系统模块，打开模块编辑框，添加相应的子系统模块，如图9.22所示。

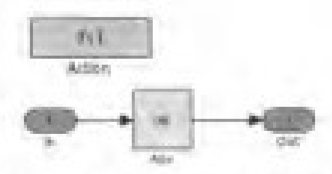


图 9.22 添加“If Action”子系统模块

step 5 设置“ElseAction”子系统模块的属性。双击“Else Action”子系统模块，打开模块编辑框，添加相应的子系统模块，如图9.23所示。

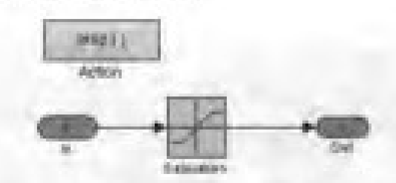


图 9.23 添加“Else Action”子系统模块

step 6 设置“Saturation”模块的属性。在上面的模块系统中，“Saturation”模块的主要功能是设置Else系统中的饱和数值范围，其属性对话框如图9.24所示。

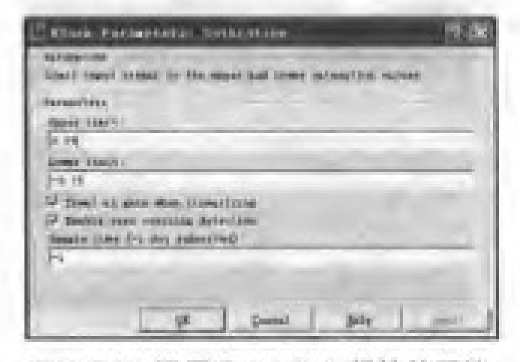


图 9.24 设置 Saturation 模块的属性

9.2.3 添加显示模块

延续上面小节的步骤。

step 1 设置"Scope"模块的属性。双击上面系统中的"Scope"模块，然后打开该模块的属性对话框，在其中设置模块的属性，如图 9.25 所示。

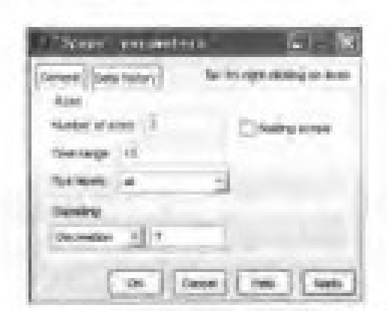


图 9.25 设置"Scope"模块的属性



在上面的属性对话框中，选中坐标轴的个数为 3，然后在"Tick labels"选框中选择"all"选项，这样 Simulink 会为每一个坐标轴添加图形的标题，该标题的具体文字就是在前面步骤中输入端口中的文字名称。

9.2.4 运行仿真系统

延续上面小节的步骤。

step 1 运行系统，得出仿真结果。前面步骤已经设置了所有的模块属性，单击"运行仿真"按钮，查看仿真结果，如图 9.26 所示。



图 9.26 查看仿真结果

step 2 查看系统模块。当完成仿真后，对应的模块系统如图 9.27 所示。

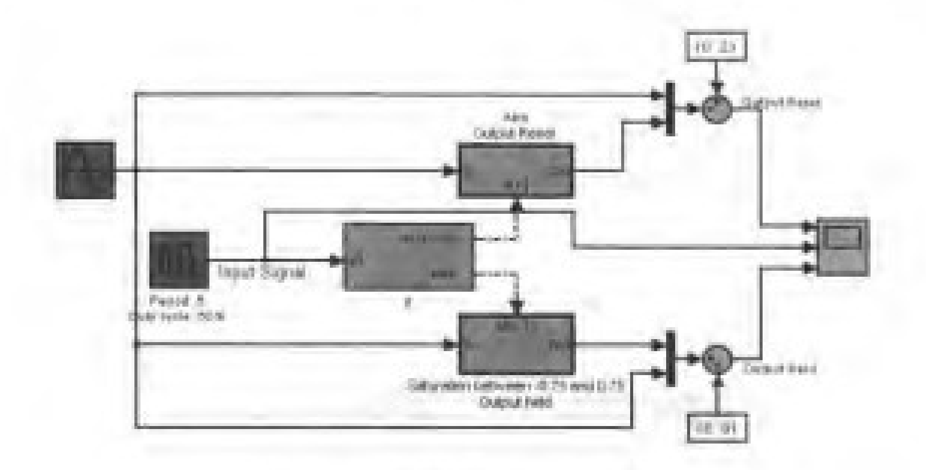


图 9.27　运行仿真后的系统模块

可以看出，当运行系统的仿真后，If 模块和 "If Action"、"Else Action" 模块之间的连线就变成了点划线，表示对应的程序模块已经运行完成。



在本实例中，基本接触到了 Simulink 中关于 "If" 子模块的基础属性，同时也演示了使用自上向下的方法创建子系统的方法，也就是首先搭建子系统模块的上层框架，然后再分别添加各个子系统的程序模块，实现模块功能。对于其他的子系统模块和创建子系统的方法将在后面章节中逐渐介绍。

9.3 封装子系统

一般（或者被称为简装）子系统操作比较简单，可以使整个模型更加简洁，也在一定程度上提高了分析研究问题时的概念抽象的能力；但是还是存在着明显的缺陷：一般子系统将直接从 MATLAB 的工作空间中获取变量数值，容易发生变量冲突，同时与 Simulink 模块库中的基础模块的结构不一致，也就是规范化的程度比较低。封装子系统（Masked subsystem）就是为了克服上面的缺点而出现的。

从外表上来看，封装子系统和普通的模块库模块完全相同，有着自己的图标、参数设置的对话框；从变量（或者被称为数据）传递的层面上来看，封装子系统有着自己的工作空间，这个工作空间将独立于基础空间和其他模块库的模块空间。

本节将通过几个比较典型的实例，介绍如何创建封装子系统，以及在创建子系统时所必须注意的事项。

9.3.1 封装子系统的创建方法

在详细介绍封装子系统的创建方法之前，首先介绍封装子系统的主要优点：

- 在设置各子系统中各模块的参数时，只需要通过参数对话框就可以完成；
- 为子系统创建一个可以反映子系统功能的图标，给用户直观的印象；
- 可以避免用户在错误操作中修改模块的参数数值。

在 Simulink 中，创建子系统的一般步骤如下：

step 1 按照前一小节中介绍的方法来创建子系统；

step 2 选择需要封装的子系统，在菜单栏中选择 "Edit" ⇨ "Mask subsystem" 命令，打开封装编辑框，如图 9.28 所示；

step 3 在封装编辑框中，设置封装子系统的参数属性、模块描述和帮助文字、自定义的图形标识等，关闭编辑器就可以得到新建的封装子系统；

step 4 如果需要编辑封装子系统，可以选中该子系统，选择 "Edit" ⇨ "Edit Mask" 命令，打开上面的对话框，重新设置相应的属性。



图 9.28　封装编辑器



封装子系统是在简装子系统基础上经过精心设置、装饰的系统，其制作过程和 Simulink 模块库中某些模块的制作方法没有太大的区别。因此了解封装子系统也可以让读者加深对模块库的了解。

9.3.2　封装子系统的步骤

在本小节中，将利用一个简单的实例来演示如何封装子系统，例子并不复杂，但是基本上涉及封装子系统的各个属性。下面分步骤详细介绍。

例 9.4　使用 Simulink 中的封装子系统工具来求解例 9.2 中的线性系统。

step 1 添加子系统的基础模块。为了在后面的步骤中设置封装子系统的属性，需要首先添加子系统的模块，如图 9.29 所示。

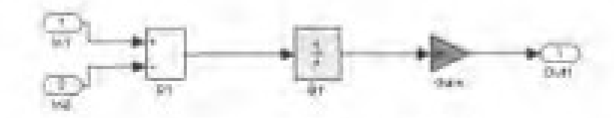


图 9.29　添加子系统的模块

step 2 设置 "Gain" 模块的属性。在例 9.2 中，变量 m 的数值是 1/2。在本实例中，为了分析封装子系统中的参数属性，将其改为 m，可以在后面的步骤中根据需要修改 m 的数值，如图 9.30 所示。

step 3 封装上面的子系统，并设置其 "图标" 属性。选中该子系统，然后选择 "Edit" ⇨ "Mask Subsystem" 命令，打开封装编辑框。选择其中的 "Icon" 选项卡，如图 9.31 所示。

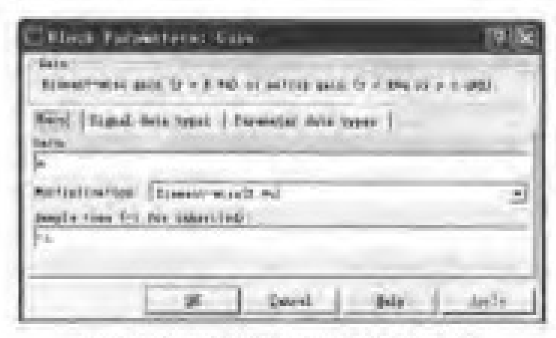


图 9.30 设置 Gain 模块的属性

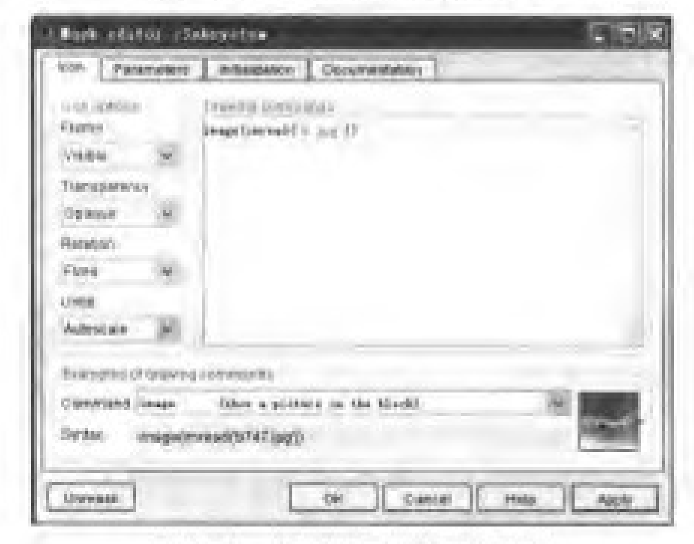


图 9.31 设置子系统的图标属性

- 在 "Icon" 选项卡的 "Icon options" 面板中，可以定义图标的边框（Frame）是否可见，生成的端口图标是否可见（Transparency），图标是否固定（Fixed）等属性，可自行尝试分析。
- 在 "Drawing commands" 选框中输入 "image(imread('9.jpg'))"，表示将在 MATLAB 的当前路径中的 "9.jpg" 文件作为封装子系统的图标。
- 在 "Examples of drawing commands" 面板中，向用户解释了如何使用各种绘制图标模块的命令，每种命令都对应在右下角有一种实例，用户可以依照该选框中的命令格式和右侧出现的示例图标来编写图标绘制命令。



在 Simulink 中，可以使用各种绘制命令来显示各种图标图案，例如 disp、dpoly、fprintf、image、patch、plotport_label 和 text 等。关于这些命令的详细使用方法，感兴趣的读者可以参看 "Simulink Reference" 中的 "Mask Icon Drawing Commands" 部分内容。

step 4 设置封装子系统的 "Parameters" 参数。选择封装编辑器中的 "Parameters" 选项卡，在其中设置封装子系统的参数数值，如图 9.32 所示。

在上面的对话框中，单击 "Add" 按钮，开始添加子系统的参数，在 "Prompt" 选框中输入 "Gain"，该字符串表示在对话框中显示的提示文字；在 "Variable" 选框中输入 "m"，表示用户在对话框中输入的数值将被传给变量 m。在 "Type" 选框中选择 "edit" 类型。最后，在 "Dialog callback" 选框中输入下面的程序代码：



图 9.32　设置封装子系统的参数

```
if str2num(get_param(gcb, 'm'))<0
    error('Gain is negative.')
end
```



在上面的程序代码中，首先检测用户输入的变量数值是否小于0；如果小于0则调用error命令，显示错误信息，同时结束程序代码。

step 5 设置封装子系统的"Initialization"属性。选择封装编辑器中的"Initialization"选项卡，在其中设置封装子系统的参数初始数值，如图 9.33 所示。

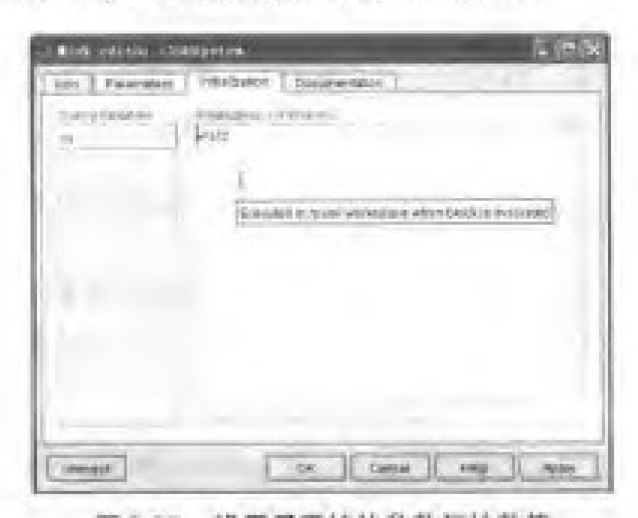


图 9.33　设置子系统的参数初始数值

在"Initialization"选项卡中，可以定义封装子系统的初始化命令。初始化的命令可以使用任何有效的 MATLAB 表达式、函数、运算符和"Parameters"选项卡汇总定义的变量，但是初始化命令不能访问 MATLAB 工作空间中的变量。



可以在该选项卡中的每一行命令的后面添加分号结束，这样就可以避免在模型仿真运行的时候在命令窗口显示仿真结果。

step 6 定义封装子系统的说明文字。选择封装编辑器中的"Documentation"选项卡，在其中设置封装子系统的说明文字，如图 9.34 所示。



图 9.34 定义封装子系统的说明文字

在上面的对话框中，可以定义封装子系统的类型、模块描述和模块的帮助信息等，这些内容可以选择性地描写，加大封装子系统的可读性。



上面对话框中设置的文字信息都会在用户打开的子系统参数对话框中显示，实现用户和系统之间的互动。

step 7 查看封装结果。上面步骤已经完成了封装子系统的属性设置，单击 "Apply" 按钮或者 "OK" 按钮，就可以保存封装结果，如图 9.35 所示。

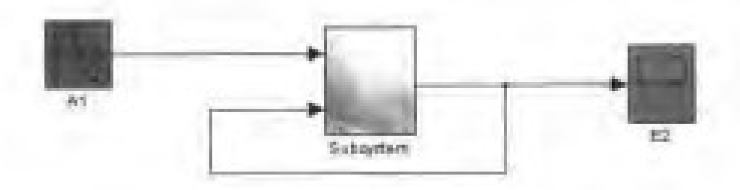


图 9.35 查看系统封装结果

step 8 设置封装子系统的参数数值。双击上面的 Subsystem 模块，打开参数对话框，在其中输入变量 m 的数值，如图 9.36 所示。



图 9.36 设置封装子系统的参数数值

在上面的对话框中，上半部分的文字就是用户在 "Documentation" 选项卡中填写的文字说明，下半部分的选框内容则是用户在 "Parameters" 选项卡中填写的参数内容，可以在该对话框中填入变量 m 的数值。

step 9 运行仿真。在设置参数后，可以运行该仿真系统，得到的结果如图 9.37 所示。

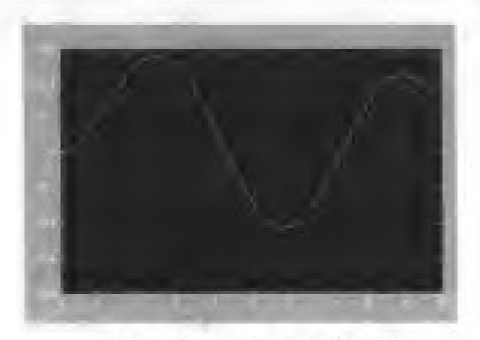


图 9.37 查看仿真结果

step 10 检测系统的错误检测功能。双击上面系统中的"Subsystem"模块，打开对应的参数对话框，在其中输入一个负数参数值 -3，然后单击"Apply"按钮，结果如图 9.38 所示。



图 9.38 系统提示错误信息



还可以根据情况来修改 Gain 的具体数值，查看系统仿真结果是否发生变化，用户可以自行分析这种情况的原因。

9.4 封装子系统实例

前面已经介绍如何在 Simulink 中创建封装子系统的方法。在本节中结合一个具体的实例，说明如何在实际中应用封装子系统。在本实例中，将会涉及该子系统的其他内容，在实例的对应位置将会给出相应的说明。由于本实例比较复杂，因此将分小节详细介绍创建该系统的步骤。

9.4.1 添加"Bang-Bang Controller"子系统

例 9.5 利用 Simulink 来模拟防锁的刹车系统（Anti-Lock Brake System），该刹车系统将根据车轮速度（Wheel Speed）和汽车角速度（Vehicle angular speed）之间的相对延迟位移（relative Sp），控制是否运行汽车的刹车系统，下面详细介绍创建步骤。

step 1 添加"Bang-Bang Controller"子系统模块。该模块的主要功能是判断输入信号是否大于 0：如果输入信号大于 0，则返回数值 1；如果输入信号小于 0，则返回数值 -1；如果输入信号等于 0，则返回数值 0，系统模块如 9.39 所示。

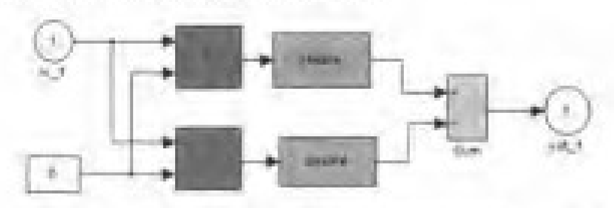


图 9.39 添加"Bang-Bang Controller"模块

step 2 封装子系统，并设置该子系统的图标。选中上面步骤中添加的系统模块，然后选择 "Edit" ⇨ "Mask Subsystem" 命令，打开封装编辑框，选择其中的 "Icon" 选项卡，在其中输入绘图命令，如图 9.40 所示。

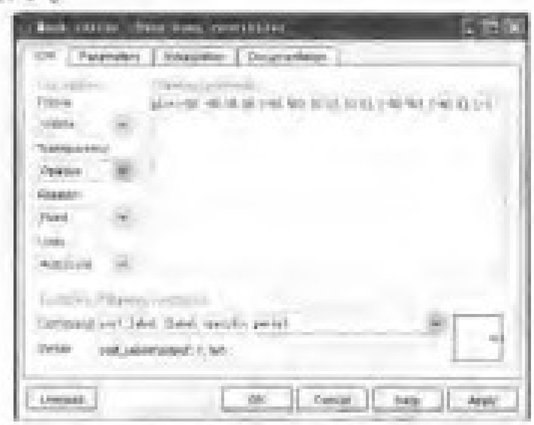


图 9.40　设置封装子系统的图标

在上面对话框中输入的绘图命令为：

```
plot([-50,-50,50,50,[-50,50],[0,0],[0,0],[-50,50],[-40,0],[-30,-30],[0,40],[30,30])
```

step 3 在命令窗口中演示上面绘图的结果是跳跃式的间断函数，可以在MATLAB的命令窗口演示该命令，得到的结果如图 9.41 所示。

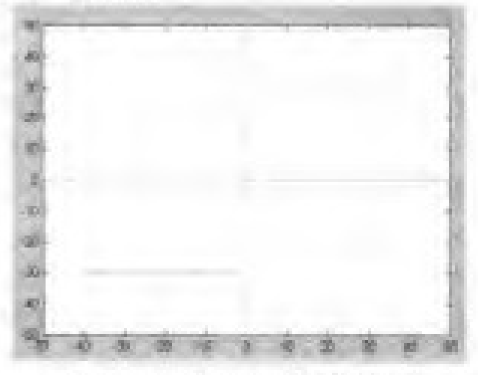


图 9.41　演示子系统的图标

step 4 设置子系统的说明文字。选择封装编辑框中的 "Documentation" 选项卡，在其中设置封装子系统的说明文字，如图 9.42 所示。

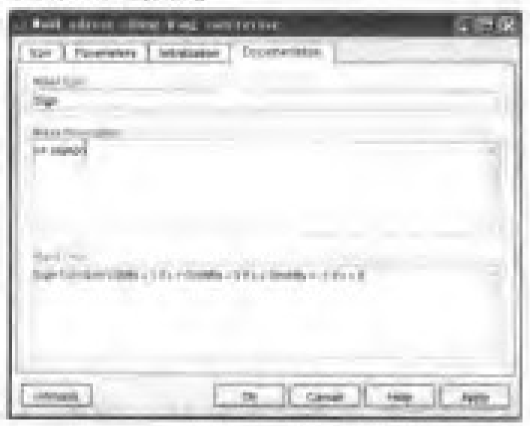


图 9.42　设置子系统的说明文字

9.4.2 添加“brake torque”子系统

延续上面小节的步骤。

step 1 保存封装子系统的属性设置，然后添加“刹车转矩（brake torque）”系统的程序模块，如图 9.43 所示。

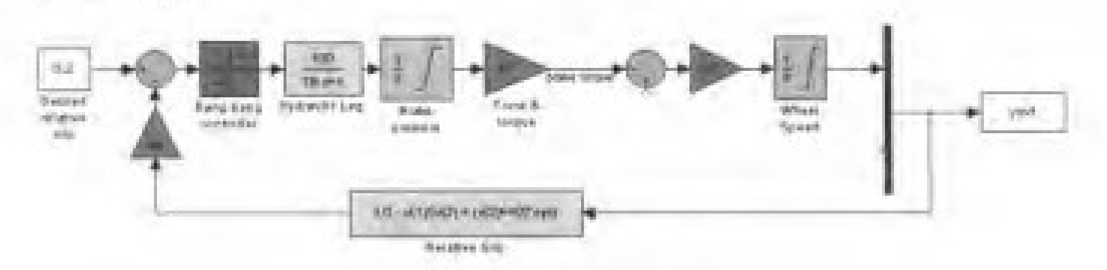


图 9.43 添加系统模块

step 2 分析系统模块的功能。上面的系统模块并不复杂，下面详细介绍各个系统模块的属性和功能。

- **“Desired relative slip”模块**：常数（Constant）模块，数值为 0.2，表示该系统自行设置的相对位移数值。
- **圆形求和（Sum）模块**：该模块主要用来计算系统设定的相对位移数值和系统积分得到的相对位移模块的数值差值。
- **“Ctrl”模块**：增益（Gain）模块，其目的在于将系统得到的相对位移进行适当的缩放，其中 Ctrl 是系统的初始参数数值。
- **“Relative Slip”模块**：该模块的主要功能在于计算整个系统的两个信号车轮速度（Wheel Speed）和汽车角速度（Vehicle angular speed）之间的相对延迟位移。其具体的表达式为“1.0 - u(1)/(u(2) + (u(2)==0)*eps)”，其中 u1 代表的是车轮速度，u2 代表的是汽车角速度，该信号将需要添加新的系统模块来进行计算。
- **“Bang-Bang controller”模块**：前面步骤中添加的封装子系统模块。
- **“Hydraulic Lag”模块**：Transfer Fcn 模块，该模块的主要功能在于将“Bang-Bang controller”模块中输出的信号通过公式转换为水力延迟数值，其中表达式中的 TB 是初始参数的数值。
- **“Brake pressure”模块**：积分器模块，将前面步骤计算得到的水力延迟数值进行数值积分，得出系统的“刹车压力”数值。
- **“Force &torque”模块**：增益模块，参数数值表示的是扭矩大小，因此，刹车压力的数值和扭矩模块相乘得到的结果就是刹车转矩（brake torque）。



关于后面的一些系统模块，由于缺乏足够的输入信号，将在后面的步骤中添加信号之后再详细介绍具体的功能。

9.4.3 添加“tire torque”子系统

延续上面小节的步骤。

step 1 添加“轮胎转矩（tire torque）”系统的程序模块，如图 9.44 所示。

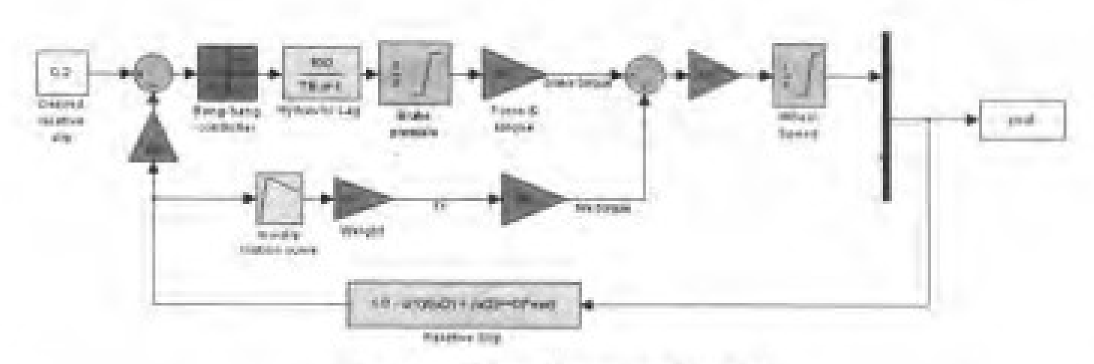


图 9.44 添加轮胎转矩的系统模块

step 2 设置 "mu-slip friction curve" 模块的属性。这个模块是 "Lookup Table" 函数模块，该模块的输入信号是计算得到的 "slip" 数值，输出的数值是系数 "mu"，也就是系统摩擦系数 μ。该模块的属性对话框如图 9.45 所示。

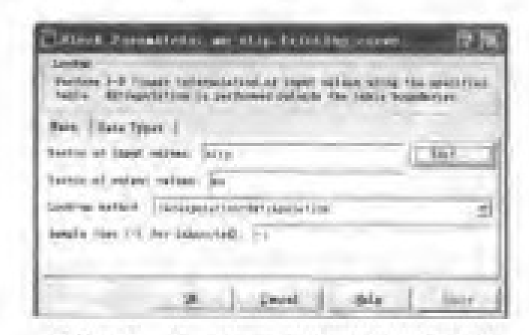


图 9.45 Lookup Table 模块的属性

step 3 分析 "mu-slip friction curve" 模块的数据。在 "Look-up method" 选框中，选择插补数据的方法为 "Interpolation-Extrapolation"，该选项是 Simulink 中的默认选项，这样系统会对数据进行线性的插补和外推。单击上面对话框中的 "Edit" 按钮，可以编辑系统的输入数据，如图 9.46 所示。

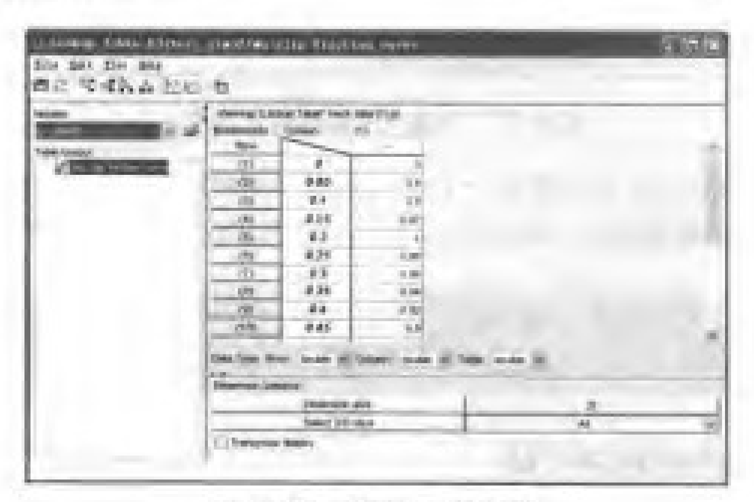


图 9.46 编辑模块的数据



"Lookup Table" 函数模块是 Simulink 中常见的非线性模块，其属于 "Lookup Tables" 模块库中。Simulink 提供多维的数据查找模块，感兴趣的读者可以查看相应的帮助文件。

step 4 设置 "Weight" 模块的属性。这个模块是 Gain 模块，该模块的功能是计算物体的重量，其表达式为 "m*g/4"，对应的参数对话框如图 9.47 所示。



图 9.47 "重量" 模块数值

step 5 设置 "Rr" 模块的属性。该模块属于 Gain 模块，该模块中的数值是 Rr，代表的是轮胎的半径。从 "Weight" 模块中输出的信号经过该模块后的结果就是 "tire torque"。



当添加上面的程序模块后，再将上面程序得出的数值和上面步骤得到的brake torque相减，得到的力距计算结果经过 "1/I" 增益模块后得到加速度，然后将经过积分器模块得到速度的数值。

step 6 添加计算汽车运动角速度的程序模块，如图 9.48 所示。

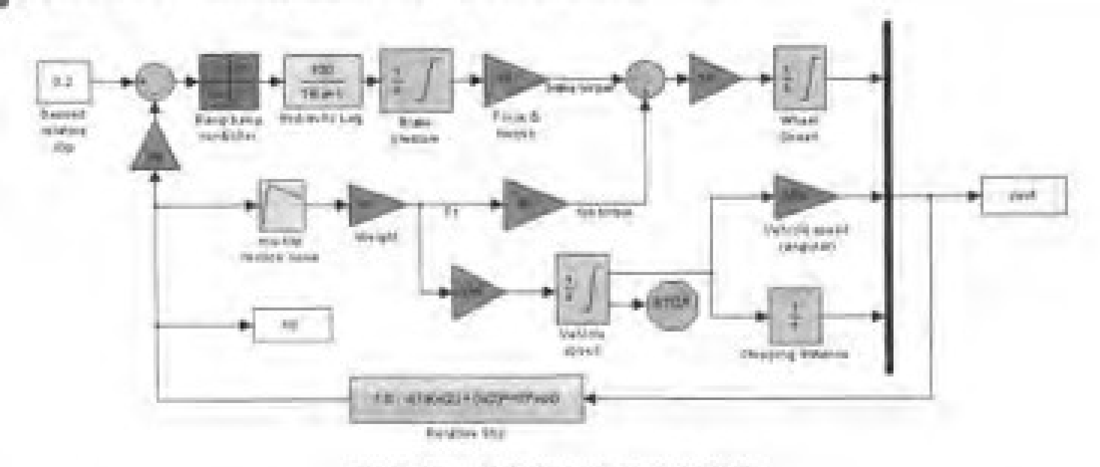


图 9.48 计算汽车运动的角速度

在上面步骤添加的模块中，首先通过 "-1/m" 增益模块得到汽车运动的加速度，将加速度经过积分器模块得到汽车运动的速度数值，最后将该数值除以汽车车轮的半径，得到汽车运动的角速度。



在上面的系统模块中，slp 和 yout 模块都是 "To Workspace" 模块，分别向 MATLAB 的工作空间输入相对位移数据和各种速度、位移数据。

9.4.4 添加子系统的程序代码

延续上面小节的步骤。

step 1 填写关于该系统的数据文件。在上面的系统模块中，引用了各种系统参数数值，在运行仿真之前必须给这些参数进行赋值。返回MATLAB的命令窗口，单击命令窗口工具栏中的□按钮，或者选择编辑栏中的 "File" ⇨ "New" ⇨ "M-file" 命令，打开 M 文件编辑器，在其中输

入下面的代码:

```
% 显示加载数据的信息
fprintf('Loading data for ABS braking model...')
% 定义常数信息
g = 32.18;
v0 = 88;
Rr = 15/12;  % Wheel radius
Kf = 1;
m = 50;
PBmax = 1500;
TB = 0.01;
I = 5;

%
% Mu slip 曲线
%
slip = (0:.05:1.0);
mu = [ 0 .4 .8 .97 1.0 .98 .96 .94 .92 .9 .88 .855 .83 .81 .79 .77 .75 .
73 .72 .71 .7];
ctrl = 1;
% 显示加载结束的信息
disp('done.');
```

将上面的所有数据保存为“Simdata.m”文件，在运行该仿真系统时，将需要首先加载上面文件中的数据。

step 2 编写运行仿真系统的代码。单击命令窗口工具栏中的 D 按钮，或者选择编辑栏中的“File”⇨“New”⇨“M-file”命令，打开 M 文件编辑器，在其中输入下面的代码:

```
% 调用仿真文件
sim32
% 设定仿真运行的时间
try
  time = sim('sim32',25);
catch
  simdata
  time = sim('sim32',25);
end
% 创建图形对象
h = findobj(0, 'Name', 'ABS Speeds');
if isempty(h),
  h=figure('Position',[ 300   387   452   257],...
           'Name','ABS Speeds',...
           'NumberTitle','off');
end
figure(h)
set(h,'DefaultAxesFontSize',8)
% 根据输出变量绘制图形
plot(time,yout(:,1:2))
% 添加图形标题和坐标轴名称
title('Vehicle speed and wheel speed')
ylabel('Speed(rad/sec)')
xlabel('Time(secs)')
% 设置图形的位置属性
set(gca,'Position',[ 0.1300    0.1500    0.7750    0.750])
```

```
%设置标题和坐标轴名称的字体大小
set(get(gca,'xlabel'),'FontSize',10)
set(get(gca,'ylabel'),'FontSize',10)
set(get(gca,'title'),'FontSize',10)

%添加带有箭头的注释文字
hold on
plot([ 5.958; 4.192],[ 36.92; 17.29],'r-',[ 5.758; 5.958; 6.029],[ 36.55;
36.92; 35.86],'r-' )
%设置文字内容
text(8.533,54.66,'Vehicle speed (\omega_v)','FontSize',10)
plot([ 7.14; 8.35],[ 43.1; 56.3],'r-',[ 7.34; 7.14; 7.07],[ 43.4; 43.1; 44.
1],'r-' )
text(4.342,15.69,'Wheel speed (\omega_w)','FontSize',10)
drawnow
hold off
%创建新的图形对象
h = findobj(0, 'Name', 'ABS Slip');
if isempty(h),
  h=figure('Position',[ 300    56    452    257],...
           'Name','ABS Slip',...
           'NumberTitle','off');
end
figure(h);
set(h,'DefaultAxesFontSize',8)
%根据变量绘制图形
plot(time,slp)
%添加图形标题和坐标轴名称
title('Slip')
xlabel('Time(secs)')
ylabel('Normalized Relative Slip')
set(gca,'Position',[ 0.1300    0.1500    0.7750    0.750])
set(get(gca,'xlabel'),'FontSize',10)
set(get(gca,'ylabel'),'FontSize',10)
set(get(gca,'title'),'FontSize',10)
```

将上面的代码保存为"runsim.m"文件，该文件将是运行该仿真的主要代码文件。关于该文件中的代码含义，请查看相应的关于图形的章节。

添加"Subsystem"子系统

延续上面小节的步骤。

step 1 添加新的"Subsystem"模块。选择"Simulink Library Browser"中"Commonly Used Blocks"模块库中的"Subsystem"模块，然后双击该模块，将子系统默认的数据输入和数据输出模块删除，得到的结果如图9.49所示。

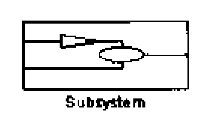


图9.49 添加新的Subsystem模块

step 2 封装上面的子系统。选中上面的子系统，单击鼠标右键，在弹出的快捷菜单中选择"Mask

Subsystem"选项，打开封装编辑器，设置子系统的图标，如图 9.50 所示。

单击上面对话框中的"Apply"按钮，保存上面的设置，得到的结果如图 9.51 所示。



图 9.50 封装子系统

Double click to run model and plot the results

Subsystem

图 9.51 设置子系统的图标

step 3 设置子系统模块的外观属性。为了设置系统模块的外观，需要设置该模块的背景颜色和名字属性等，得到的结果如图 9.52 所示。



在上面的步骤中，将子系统模块的名称首先改为"Run"，之所以然后将其隐藏，是因为该模块上的文字已经介绍了该模块的功能。

step 4 设置模块的属性。选择该系统模块，单击鼠标右键，在弹出的快捷菜单中选择"Block Properties"选项，打开属性对话框，选择"Callbacks"选项卡，设置"OpenFcn"对应的调用函数，如图 9.53 所示。

Double click to run model and plot the results

图 9.52 设置系统模块的外观属性

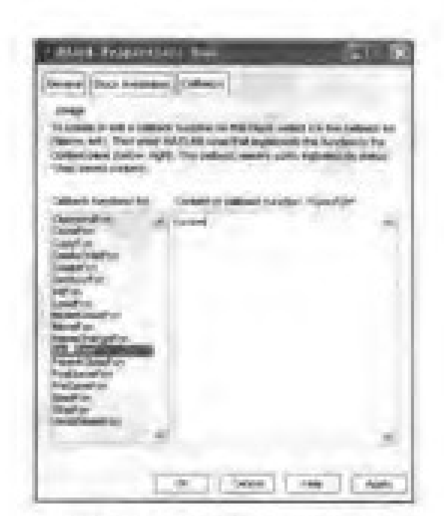


图 9.53 设置模块的属性

在上面的对话框中，将该模块的"OpenFcn"的调用函数设置为前面步骤中编写的"runsim"，当双击该模块时，Simulink 将会调用 runsim 的 M 文件。



在对话框中的"Callback function list"选框中列出了所有相关的调用函数名称，如果在右侧的选框中编写了对应的调用函数名称，则会在列表名称后面添加*号。

step 5 查看添加的所有系统模块。前面的步骤已经添加了所有的系统模块，用户可以查看完成的完

整系统模块，如图 9.54 所示。

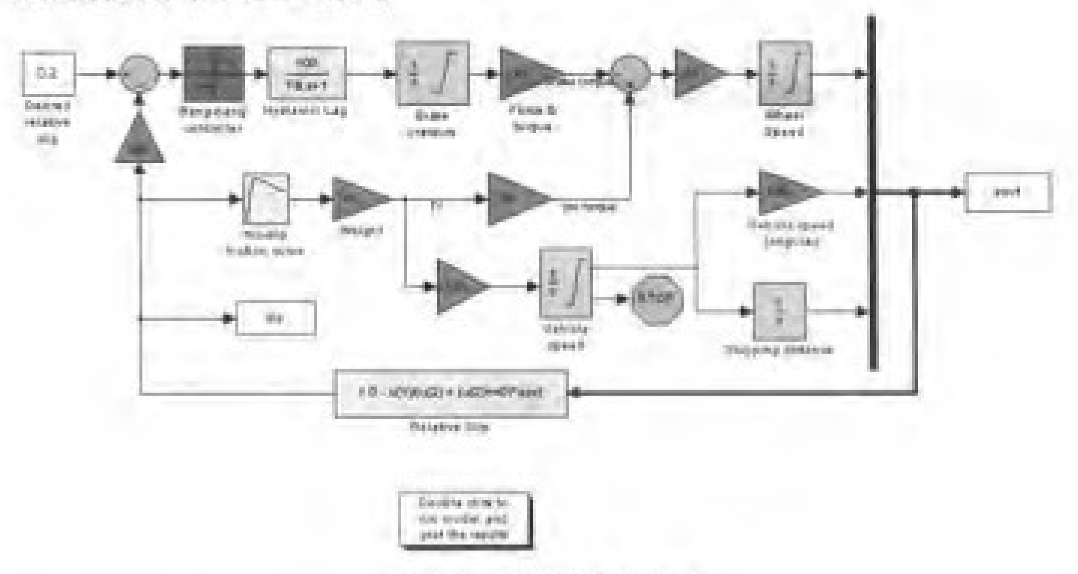


图 9.54　完整的系统模块

9.4.6 运行仿真系统

延续上面小节的步骤。

step 1 运行仿真模块。在本实例中，用户将不能使用 Simulink 中的仿真按钮得到最后的结果，而需要双击前面步骤中添加的封装子系统模块，得到的结果如图 9.55 所示。

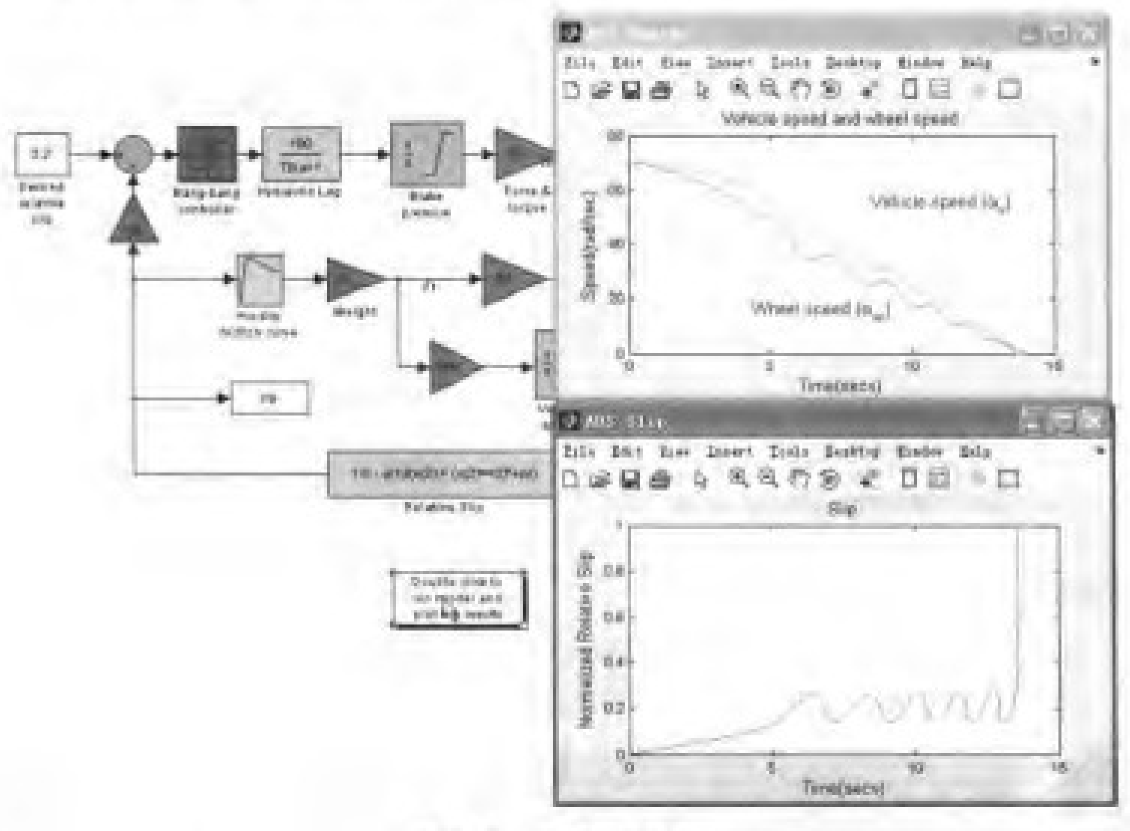


图 9.55　查看仿真结果

step 2 查看 MATLAB 命令窗口中的数据。返回 MATLAB 的命令窗口，查看 Simulink 运行仿真后的命令窗口，如图 9.56 所示。

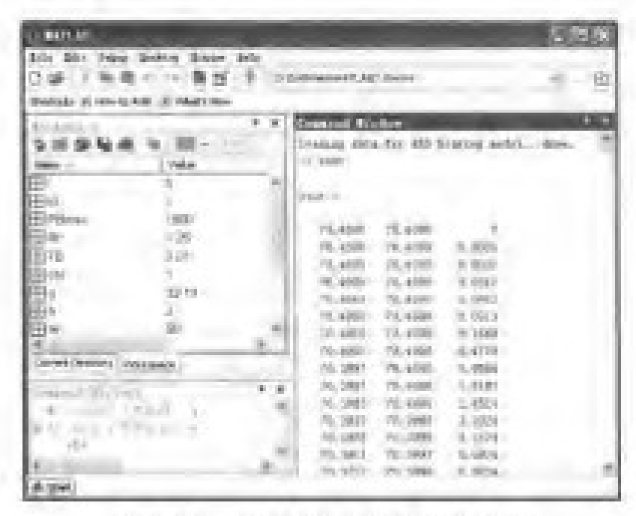


图 9.56 返回 MATLAB 命令窗口

在 MATLAB 的命令窗口中，选择 "Workspace" 选项卡，查看 MATLAB 的工作空间中的变量情况，表明通过 Simulink 来加载原始数据成功，同时，在命令窗口中显示加载信息：

```
Loading data for ABS braking model...done
```

还可以查看仿真后的变量数据。例如，当输入 "yout"，查看最后的输出结果，得到的结果如下：

```
>> yout
yout =
   70.4000   70.4000         0
   70.4000   70.4000    0.0000
   70.4000   70.4000    0.0002
   70.4000   70.4000    0.0012
   70.4000   70.4000    0.0062
   ............................................//限于篇幅，省略了部分数据
         0    0.6787  720.6747
         0    0.6337  720.6829
         0    0.5886  720.6906
         0    0.5436  720.6977
         0    0.4985  720.7042
         0    0.4535  720.7101
         0    0.4084  720.7155
         0    0.3634  720.7203
         0    0.3183  720.7246
         0    0.2733  720.7283
         0    0.2282  720.7314
         0    0.1832  720.7340
         0    0.1381  720.7360
         0    0.0931  720.7375
         0    0.0480  720.7383
         0    0.0030  720.7387
         0    0.0000  720.7387
         0         0  720.7387
```



除了可以显示仿真系统得出的结果之外，还可以对仿真结果进行多种处理。限于篇幅，就不详细介绍了，请读者自行尝试。

9.5 使能（Enabled）子系统

在Simulink中，使能（Enabled）子系统是条件执行子系统中的一种，只有当控制信号为正值的时候，才会在仿真时间步长时执行子系统。一般来讲，使能子系统包含一个控制信号的输入数值，该控制信号数值可以是标量也可以是向量。当该子系统的输入变量是标量且该输入数值大于0时，Simulink将执行该子系统；当子系统的输入变量是向量且向量中任何一维数据大于0时，则执行该子系统。

Simulink将会使用"zero-crossing slope"方法，判断是否应该执行某个使能子系统。如果信号穿过零点并且斜率大于0，则执行该使能子系统；如果信号穿过零点并且斜率小于0，系统则禁止该使能子系统。下面分小节详细介绍关于使能子系统的相关知识。

9.5.1 创建使能子系统

在Simulink中提供专门的"Enabled Subsystem"模块，可以使用该模块来创建使能子系统。创建方法和其他子系统并没有太大的不同，但是关于使能子系统Simulink提供一些比较特殊的属性选项，本小节将利用一个简单的例子来说明如何创建使能子系统。

例 9.6 利用四个离散模块和一个控制信号，演示Simulink中使能子系统的工作原理。

step 1 分析系统模块，在本系统中包含的四个离散模块如下：

- **模块 A**：Unit Delay模块，样本时间为0.25秒。
- **模块 B**：Unit Delay模块，样本时间为0.5秒。
- **模块 C**：包含在"enabled subsystem"中，Unit Delay模块，样本时间为0.125秒。
- **模块 D**：包含在"enabled subsystem"中，Unit Delay模块，样本时间为0.25秒。



之所以在该系统中添加这些系统模块，是为了系统信号在时间上的错开，这样就可以在后面的步骤中显示使能子系统的启动时间。

step 2 添加系统模块，得到的结果如图9.57所示。

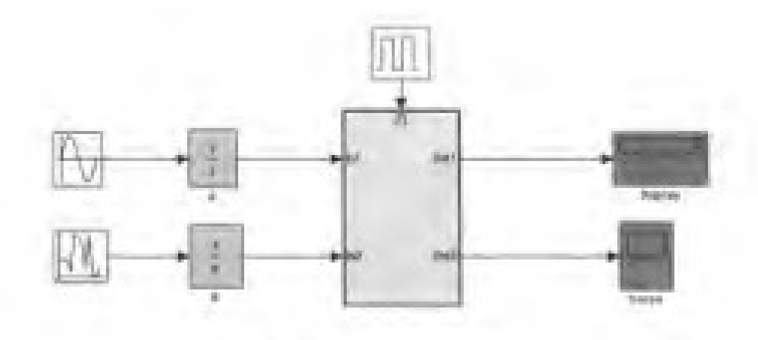


图9.57 添加系统模块

step 3 设置方形波属性。在上面的系统中，控制信号是方形波，该方形波的数值在 0.375 秒从 0 变到 1，在 0.875 秒从 1 变到 0，该模块的属性对话框如图 9.58 所示。



图 9.58 方形波的属性

step 4 添加子系统模块。双击上面的使能子系统，在打开的模块编辑框中添加子系统的模块，如图 9.59 所示。

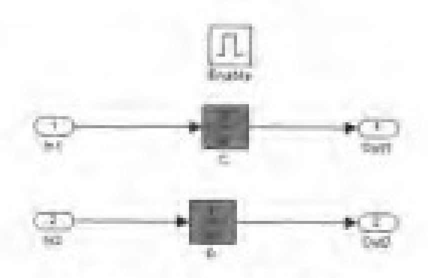


图 9.59 添加子系统模块

step 5 设置"Enable"模块的属性。双击"Enable"模块，设置模块对应的属性，如图 9.60 所示。

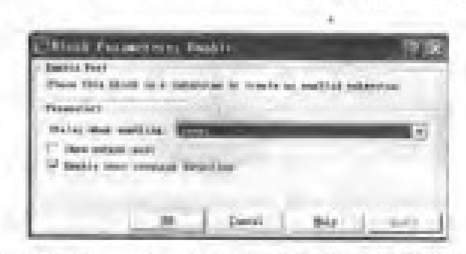


图 9.60 设置"Enable"模块的属性

step 6 运行系统仿真，得到仿真结果。将仿真时间设置为 20，单击模块对话框中的"开始仿真"按钮，运行系统仿真，得到仿真结果，如图 9.61 所示。

上面的仿真结果并不复杂，在这里就不详细说明其具体含义了。为了帮助读者了解整个仿真的运行情况，图 9.62 显示了使能子系统的启动时间。

9.5.2 使能子系统实例

上面介绍了关于使能子系统的基础知识，在本小节中将使用一个综合实例来说明如何设置使能

子系统的属性，产生不同效果的信号。

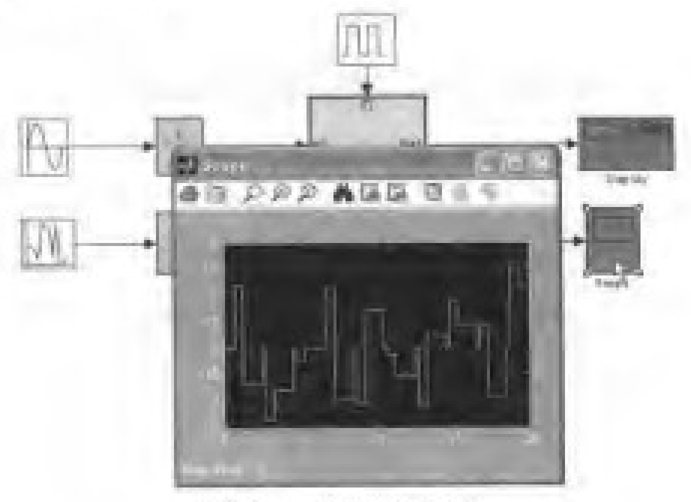


图 9.61　得到仿真结果

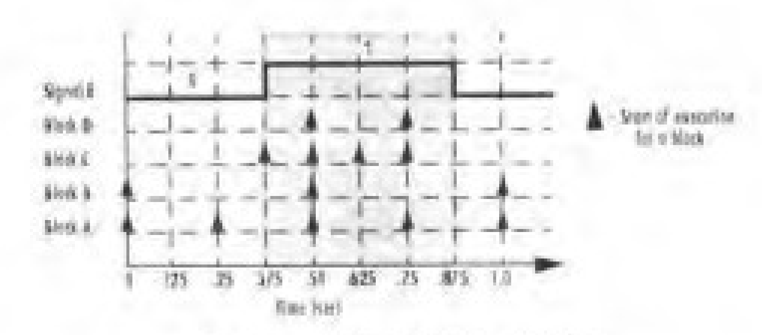


图 9.62　使能子系统的运行情况

例 9.7　创建一个使能子系统模块，该子系统根据不同的属性对同一个输入信号和控制信号产生不同的输出信号结果。

step 1 添加系统模块。根据本实例的要求，可以添加系统模块，如图 9.63 所示。

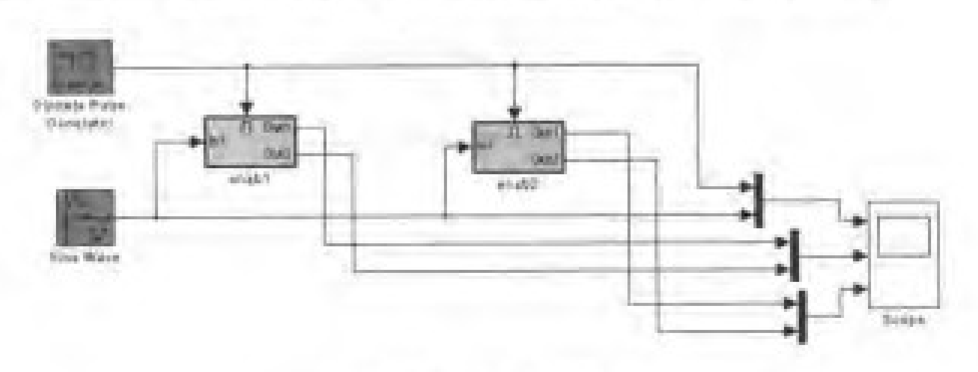


图 9.63　添加系统模块

step 2 设置输入信号的属性。双击"Sine Wave"模块，打开对应的属性对话框，在其中设置输入信号的属性，如图 9.64 所示。

step 3 查看输入信号的图形。根据上面对话框中的属性，"Sine Wave"模块将会产生离散的正弦函数图形。如果不太熟悉离散函数的图形特性，可以在系统运行之前首先查看函数图形，如图 9.65 所示。



图 9.64 设置输入信号的属性



图 9.65 查看离散函数的图形

step 4 设置控制信号的属性。双击 "Discrete Pulse Generator" 模块，打开对应的属性对话框，在其中设置控制信号的属性，如图 9.66 所示。



图 9.66 设置控制信号的属性



之所以在前面的步骤中将输入信号和控制信号都设置成离散形式，是为了在后面的步骤中比较不同的信号特征。

step 5 添加子系统的模块。双击上面系统中的"enabl"模块，在打开的模块编辑器中添加对应的子系统模块，如图 9.67 所示。

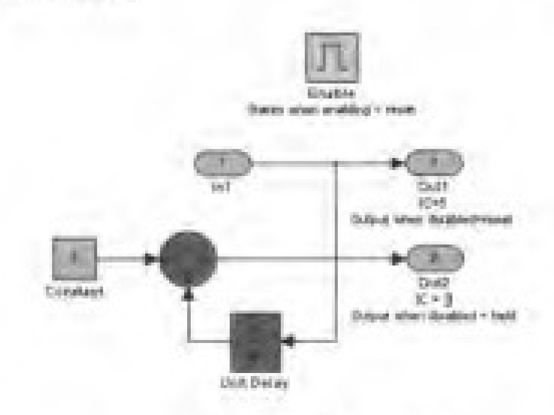


图 9.67　添加子系统模块

step 6 添加子系统的"Enable"模块属性。双击上面的"Enable"模块，打开对应的属性对话框，设置该模块的属性，如图 9.68 所示。



图 9.68　设置"Enable"模块的属性

在上面的对话框中，将"States when enabling"选框的属性设置为"reset"，表明当系统启动"使能"子系统模块时，将会把所在子系统所有内部状态重新设置为初始数值。

step 7 设置子系统中"Out1"模块的属性。选择上面子系统中的"Out1"模块，单击鼠标右键，在弹出的快捷菜单中选择"Outport Parameters"选项，打开对应的属性对话框，在其中设置该模块的属性，如图 9.69 所示。



图 9.69　设置输出模块的属性

在上面对话框中的"Output when disabled"选框中选择"Reset"，该选项属性表明当系统中的"使能"子系统没有运行的时候，输出信号使用的是"Reset"状态。同时，设置该输出模块的初始数值为 5，当系统恢复初始状态时，将保持数值为 5。



在上面的系统模块中，当运行使能子系统时，"使能"状态数值为"Reset"；但是，仅仅设置这个属性选项还不能确定输出信号的属性。因此，在本步骤中，有必要设置输出模块的信号输出的属性。

step 8 设置子系统中"Out2"模块的属性。选择上面子系统中的"Out1"模块，单击鼠标右键，在弹出的快捷菜单中选择"Outport Parameters"选项，打开对应的属性对话框，在其中设置该模块的属性，如图 9.70 所示。

图 9.70　设置输出模块 2 的属性

在上面对话框中的"Output when disabled"选框中选择"held"，该选项属性表明当系统中的"使能"子系统没有运行的时候，输出信号使用的是"held"状态。同时，设置该输出模块的初始数值为"[]"，当系统恢复初始状态时，将保持数值为 0。

step 9 添加第二个子系统的模块。双击上面系统模块中的"enab2"模块，打开模块编辑器，在其中添加该系统的模块，如图 9.71 所示。

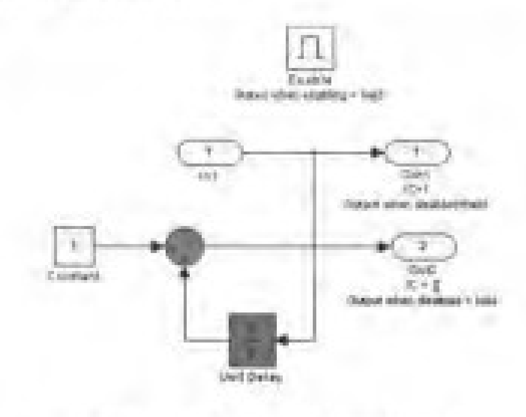


图 9.71　添加第二子系统的模块

上面的子系统模块和第一个子系统模块类似，只是在属性上有点差别。其中，"Enable"模块的状态属性为"held"；第一个输出模块的状态属性也是"held"，同时初始数值为 1；第二个输出模块的状态属性是"held"，初始数值为 0。



关于上面各个模块的具体设置情况在这里就不重复介绍了，读者可以参看前面步骤的设置方法。

step 10 运行系统仿真。单击模块编辑器中的"开始仿真"按钮，运行整个系统，得到的结果如图 9.72 所示。

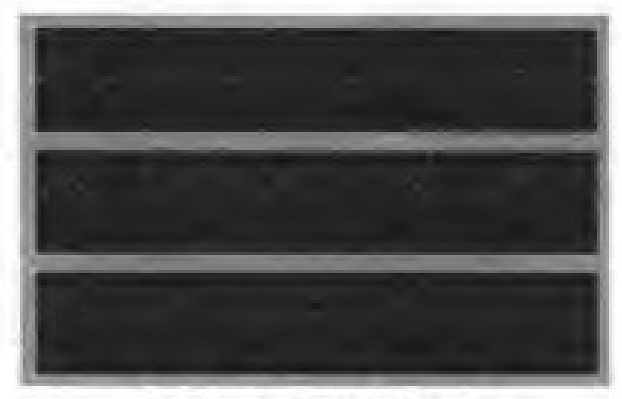


图 9.72 查看系统仿真的结果



从上面的结果来看，尽管输入信号和控制信号完全相同，但是当为"Enable"模块和输出模块设置不同的状态属性的时候，得出的结果会完全不同。

9.6 触发（Triggered）子系统

在 Simulink 中，触发子系统是条件执行子系统中的一种，该子系统只有当触发事件（或者是信号）发生的时候才会执行。一个典型的触发子系统中包括信号控制输入端口，也被称为触发输入端口，这个端口输入的信号将决定执行子系统的条件。

9.6.1 触发子系统简介

可以设置三种类型的触发事件来触发不同的子系统，各种触发事件的情况如下。

- rising：当控制信号从负数或者零转换到正数，或者当初始数值为负时，信号转换到零的时候，将会触发该子系统。
- falling：当控制信号从正数或者零转换到负数，或者当初始数值为正时，信号转换到零的时候，将会触发该子系统。
- either：当系统发生上面两种情况中任何一种情况，都会触发子系统。



如果在离散系统中使用了触发模块，只有当信号在上升或者下降之前，多于单位时间步的时间范围内保持数值0，才会触发子系统，这样就可以避免因为系统采样而触发子系统。

为了演示上面的提示信息，可以查看离散系统的时间图表，如图 9.73 所示。

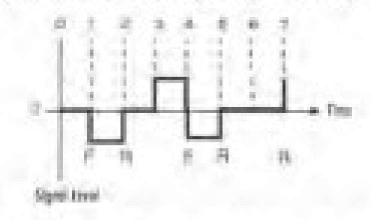


图 9.73 离散系统的时间图表

在上面的图表中，在时间 t=3 的时候，rising(R) 触发系统不会发生，因为，在该时间步信号上升之前，仅保存一个时间段的零值，因此不能触发子系统。

9.6.2 触发子系统的属性

在本小节中，将以一个比较简单的实例来介绍触发子系统的触发属性，为了让读者了解详细的属性内容，将分步骤详细介绍。

例 9.8 创建一个简单的触发子系统模块，使用不同的触发类型，得出不同的输出信号。

step 1 分析系统的输入信号。在本实例中，将选择频率为 8rad/s 的正弦波作为系统的输入信号，其对应的属性如图 9.74 所示。



图 9.74　系统输入信号的属性



在上面的系统模块中，为了方便在后面的步骤中分析子系统的性能，需要显示该波形的主要属性——频率。在其模块属性对话框中的“Block Annotation”选项卡中添加“Freq: %<Frequency> rad/sec”的描述字符，其语法在前面介绍过，这里就不详细介绍了。

step 2 分析系统的控制信号。在本实例中，将选择频率为 1Hz 的方形波作为系统的控制信号，其对应的属性如图 9.75 所示。



图 9.75　系统控制信号的属性

在上面对话框中的“Wave form”选框中选择“square”类型，表示控制信号类型为方形波，

同时在 "Freqency" 选框中输入 "1"，在 "Units" 选框中选择 "Hertz" 作为频率单位，因此也就设置了该方形波的频率为 1Hz。



在实例中，输入信号是连续的正弦函数图形，控制信号则是离散的方形波图形，这样设置的目的在于在后面步骤中更方便地分析触发子系统的性能。

step 3 添加系统模块。根据前面步骤设置的输入信号和控制信号，可以添加对应的系统模块，如图 9.76 所示。

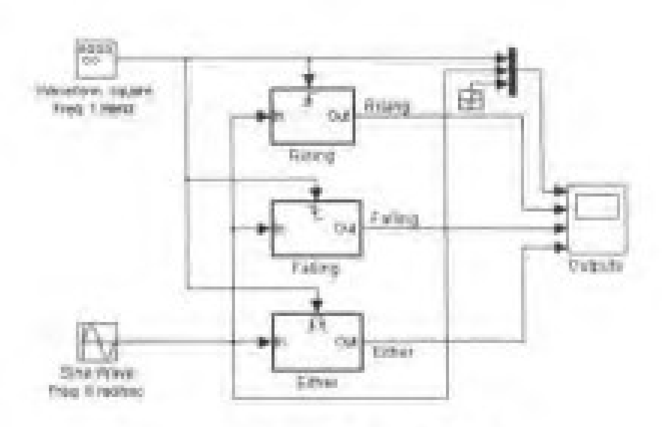


图 9.76　添加系统模块



在上面的系统模块中总共有三个触发子系统，分别是三种不同的触发类型，从上向下依次为 Rising、Falling 和 Either。这三种触发机制对应的输入信号和控制信号都是相同的，因此可以比较因为出发机制不同得出的不同结果。

step 4 运行系统仿真。当添加所有模块后，运行系统仿真，结果如图 9.77 所示。

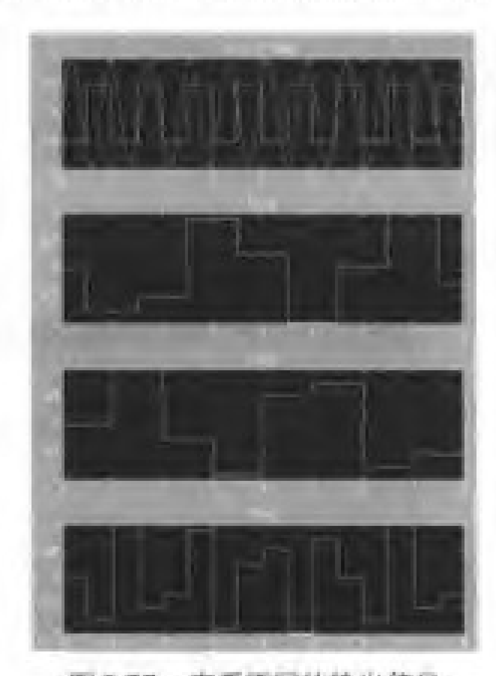


图 9.77　查看不同的输出信号

在上面的图形结果中，第一幅图形显示输入信号和控制信号，第二幅图形显示Rising触发子系统的输出结果，第三幅图形显示Falling触发子系统的输出结果，第四幅图形显示Either触发子系统的输出结果。

从上面的结果中可以看出，在Simulink中，触发子系统一旦被触发，其输出结果就会保持不变，直到下次系统被再次触发为止。在Simulink中，触发子系统的输入信号也可以是向量。如果是向量输入信号，则向量中的一个分量信号触发了系统，则子系统就会被触发。



在Simulink中，还可以使用function-call函数的方式来调用触发子系统，由于这种触发方法必须和S函数配合使用，在这里就不详细介绍了。

9.7 触发子系统实例

本节将使用一个比较综合的实例，介绍如何使用触发子系统来仿真某机械从启动到均衡速度的过程。该过程将涉及到节流系统（Throttle）、吸气系统（Intake）、压缩系统（Compression）、燃烧系统（Combustion）和机械系统（Vehicle Dynamics）等。在运行的整个过程中，其中机械最终的运行速度将会影响整个系统的气阀吸气速度，从而制约该综合系统的压缩系统（Compression）。在Simulink中，该部分的功能将需要通过触发系统实现，由于整个系统比较复杂，下面将分小节详细介绍该系统的实现过程。

9.7.1 添加系统模块

例 9.9 创建仿真系统实现某机械的发动机从启动到真正运行阶段的物理系统。

step 1 添加整个机械的发动机运行模块，得到的结果如图 9.78所示。

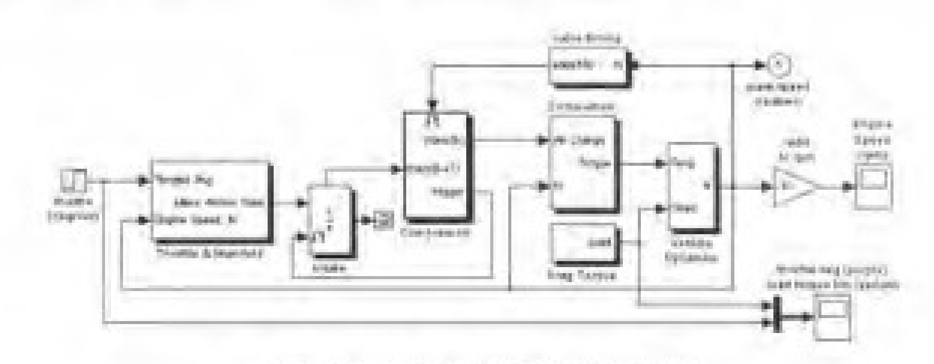


图 9.78 整个机械运行的系统模块

在上面的系统模块中，为了简化用户分辨整个系统中的不同模块，对包含子系统的模块都设置了模块阴影，因此，包含子系统的模块有："Throttle & Manifold"、"Intake"、"Compression"、"valve timing"、"Combustion"、"Drag Troque" 和 "Vehicle Dynamics" 等，这些模块分别代表整个系统中的某个部分，分别实现对应的系统功能。



由于整个系统比较复杂，在这里将无法完全按照步骤来介绍该系统的搭建过程，因此，在本小节中将介绍该系统中比较重要的模块属性。

step 2 设置 "throttle" 模块的属性。在本模块系统中，"throttle" 模块的功能是输出调节阀的角度，从属性角度来看属于 "阶跃" 波形。该阶跃波形在的时刻从初始数值 8.973 跳跃到 11.93，然后保持不变，其对应的属性对话框如图 9.79 所示。



图 9.79　设置输出信号的属性



在比较复杂的系统中，系统模块的初始数值是比较重要的参数，这些数值将直接影响整个系统的运行情况。

9.7.2　设置 "Throttle & Manifold" 子系统属性

延续上面小节的步骤。

step 1 设置 "Throttle & Manifold" 子系统的模块。双击上面系统中的 "Throttle & Manifold" 模块，在打开的模块编辑器中，添加对应的系统模块，如图 9.80 所示。

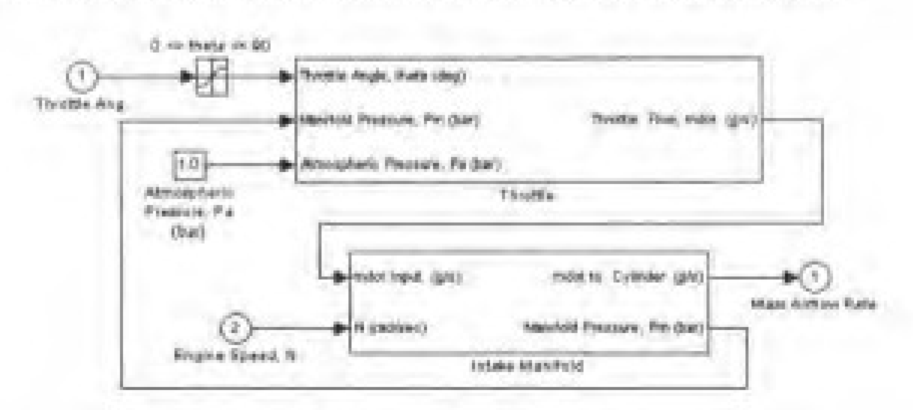


图 9.80　设置 "Throttle & Manifold" 子系统模块

分析子系统模块的功能。上面的子系统模块的功能是通过两个输入信号变量 Throttle Angle 和 Engine Speed，经过必要的模块运算，得到输出变量 "Mass Airflow Rate"。其具体的转换方式为：通过其中的子系统将变量 Throttle Angle（气压阀角度）、Manifold Pressure（支管压力）和 Atmospheric Pressure（大气压力）通过计算得到变量 Throttle Flow（气压流体），然后将计算得到的变量和 Engine Speed（发动机速度）通过下一个子系统模块得出输出变量 mdot to Cylinder（汽缸速度）和 Manifold Pressure（支管压力）。

step 3 设置 "Saturation" 模块的属性。"Saturation" 模块的目的是为了控制输入信号的角度范围，其属性如图 9.81 所示。

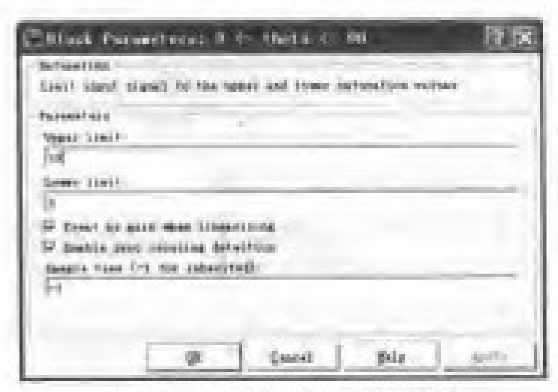


图 9.81　设置控制信号的属性

step 4 添加计算变量 Throttle Flow（气压流体）的子系统模块。双击上面系统的 Throttle Flow（气压流体）的模块，在其中添加子系统的模块如图 9.82 所示。

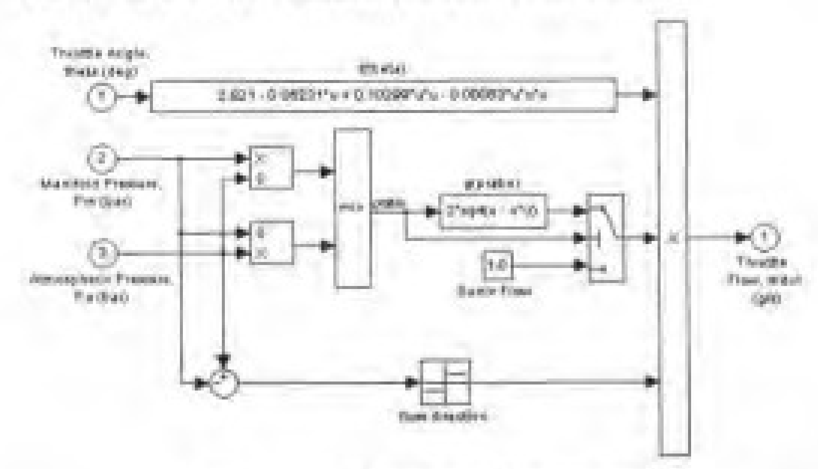


图 9.82　计算中间参量的子系统模块

step 5 分析上面子系统模块的功能。在上面的子系统模块中，首先将调节阀的角度 θ 通过经验公式 $f(\theta)$ 计算出中间参数数值 δ_1，其中 $f(\theta)$ 的具体表达式为：$2.282-0.05231\theta+0.10299\theta^2-0.00063\theta^3$；然后计算变量 Manifold Pressure（支管压力）和 Atmospheric Pressure（大气压力）的数值比例 λ，通过程序模块保证 $\lambda<1$，然后当 λ 小于 0.5 时，中间参数数值 δ_2 就等于常数 Sonic Flow；当 λ 大于 0.5 时，将其通过公式 $2\sqrt{\lambda-\lambda^2}$ 计算得到中间参数数值 δ_2；最后，通过"signum"模块计算流体压力的方向，当大气压力大于支管压力的时候，选择正号；当大气压力小于支管压力的时候，选择负号。

在上面子系统的最后，将两个中间参数相乘得到变量 Throttle Flow（气压流体）的数值，上面的计算公式相当于下面的公式：Throttle−Flow=signum$(Pm-Pa)\times\delta_1\times\delta_2$。



在上面的分析过程中，多次出现经验公式来计算某些物理参量。至于这些经验公式的由来和具体的参数含义，由于过于专业，这里就不详细介绍了，读者也没有太大的必要了解具体的含义。之所以在这里详细展开分析具体的含义，是为了读者更加熟悉模块的功能。

step 6 添加计算输出变量 mdot to Cylinder（汽缸速度）和 Manifold Pressure（支管压力）子系统模块，添加的系统模块如图 9.83 所示。

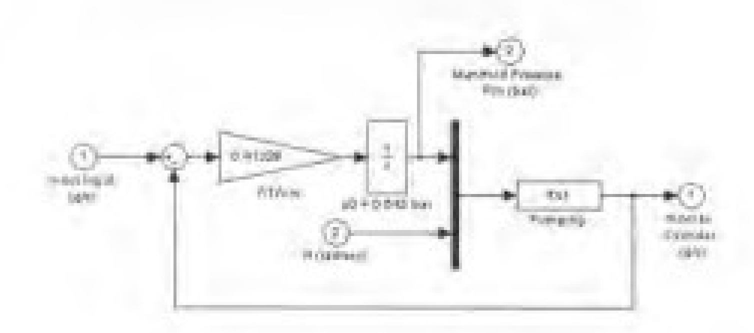


图 9.83 计算中间参数的子系统模块

step 1 分析上面子系统模块的功能。上面的子系统模块并不复杂，假定Manifold Pressure（支管压力）为变量 x，mdot to Cylinder（汽缸速度）为变量 y，上面的系统相当于求解下面的方程组：

$$\begin{cases} 0.08979vx' - 0.0337v(x')^2 + 0.0001v^2x' = y \\ 0.41328(T - F - y) = x' \\ x'(0) = 0.543 \end{cases}$$

在上面的方程组中，代表系统最后求得的发动机速度，$T-F$就是在上面子系统模块中求得的变量Throttle-Flow 数值，其中 0.41328 是量纲转换系数。



由于整个系统比较复杂，在求解某个变量的时候，除了需要了解各个变量的物理含义之外，还需要了解各个变量的物理单位。在不同量纲的物理量之间进行转换的时候，有时必须注意变量纲转换的系数问题。

9.7.3 设置“Intake”子系统属性

延续上面小节的步骤。

step 1 设置“Intake”模块的属性。双击上面系统中的“Intake”模块，设置该积分的重设属性，如图 9.84 所示。



图 9.84 设置积分器模块的属性

step 2 分析"Intake"模块的属性。在上面的对话框中，将积分器的"External reset"属性设置为"either"，该积分器的主要功能是将"Throttle & Manifold"子系统模块输出的中间变量进行积分，得到的结果是整个系统吸入的气体质量。这一点可以从量纲的角度分析，变量 Mass Airflow Rate 的单位是 g/s，经过积分得到的物理量的单位是 g。



由于所有的变量都是中间参量，因此在系统中使用的是该积分器模块的状态端口，而不是输出端口。因此为了避免输出端口悬垂，添加了"Terminator"模块。

9.7.4 设置"Compression"子系统属性

延续上面小节的步骤。

step 1 设置"Compression"子系统模块的属性。双击上面系统中的"Compression"模块，添加对应的系统模块，如图 9.85 所示。

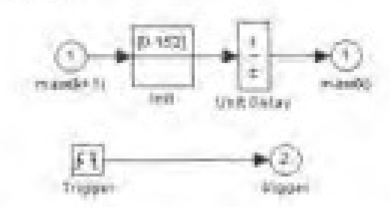


图 9.85 "Compression"子系统模块

在上面的模块中，将积分器模块中输出的"质量"变量值"mass(k+1)"输入该子系统中，经过 Unit Delay 模块得到压缩后的气体质量"mass(k)"。

step 2 设置"Trigger"模块的属性。双击"Trigger"模块，打开模块的属性对话框，设置整个系统的触发类型，如图 9.86 所示。

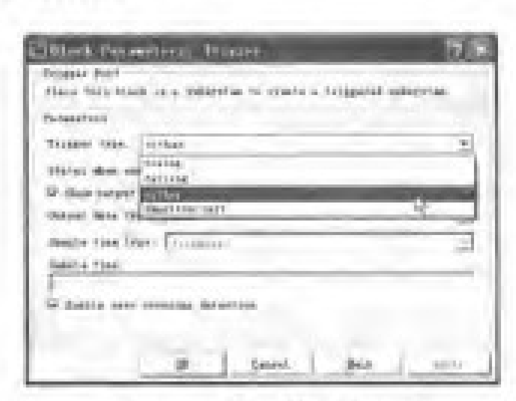


图 9.86 设置触发器类型

9.7.5 设置"Combustion"子系统属性

延续上面小节的步骤。

step 1 设置"Combustion"子系统的模块。双击上面系统中的"Combustion"模块，添加对应的系统模块，如图 9.87 所示。

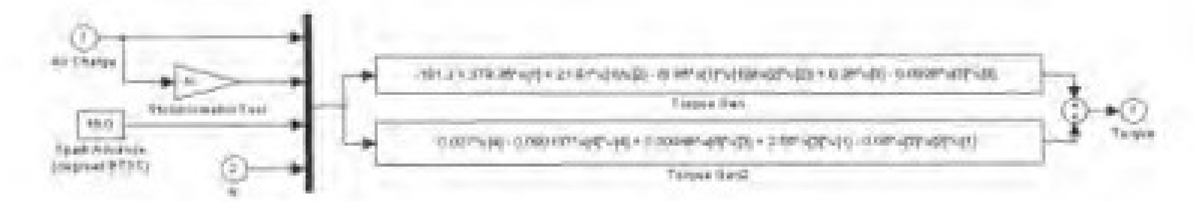


图 9.87 设置"Combustion"子系统模块

step 2 分析"Combustion"子系统的功能。在上面的系统模块中，首先将变量 Air Charge、Stoichiometric Fuel（化学燃料）和 spark advance（进给）通过下面的经验公式得到发动机扭矩的第一部分数据 δ_1：

$$-181.3+379.36x_1+21.91\frac{x_1}{x_2}-0.85\left(\frac{x_1}{x_2}\right)^2-0.26x_3-0.0028x_3^2$$

然后将变量 Air Charge、spark advance（进给）和发动机速度通过下面的经验公式得到发动机扭矩的第二部分的数据 δ_2：

$$0.027x_4-0.000107x_4^2+0.00048x_3x_4+2.55x_1x_3-0.05x_1x_3^2$$

最后，通过加法模块将上面两个部分的数据相加得到发动机的扭矩数值

$$\delta=\delta_1+\delta_2$$



在上面的公式中，变量 Stoichiometric Fuel（化学燃料）和 spark advance（进给）都是常数变量模块，而另外两个变量则是经过系统运算得到的中间变量。

9.7.6 设置"Drag Torque"子系统属性

延续上面小节的步骤。

step 1 设置"Drag Torque"子系统的模块。双击上面系统中的"Drag Torque"模块，添加对应的系统模块，如图 9.88 所示。

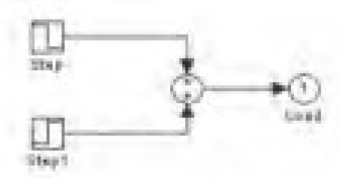


图 9.88 设置"Drag Torque"子系统的模块

在上面的模块中，将输出该系统的阻力矩的数值，其由两种阶跃波形相互叠加而得到，这是一种简化的阻力矩，两种阻力矩的主要参数分别如下。

- 跳跃时间：2s；初始数值：25；终止数值：20。
- 跳跃时间：8s；初始数值：0；终止数值：5。

step 2 显示叠加信号的结果。为了让读者有一个更加完整的印象，可以使用 Scope 模块来显示两个信号叠加后的结果，如图 9.89 所示。



图 9.89 阻力矩的信号图形

9.7.7 设置“Vehicle Dynamics”子系统属性

延续上面小节的步骤。

step 1 设置“Vehicle Dynamics”子系统的模块。双击上面系统中的“Vehicle Dynamics”模块，添加对应的系统模块，如图 9.90 所示。

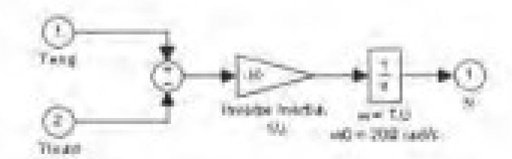


图 9.90 设置“Vehicle Dynamics”子系统的模块

step 2 分析“Vehicle Dynamics”子系统的功能。在上面的系统模块中，将前面步骤中计算得到的发动机扭矩和阻力矩相减，得到发动机实际承受的动力矩，然后将其和逆惯性矩相乘再经过积分，得到发动机的速度 N。



根据量纲分析，经过该系统得到的发动机速度的量纲是 rad/s，也就是弧度每秒，但是本实例中需要得到的单位是 rpm，因此还需要经过量纲的转换工作。

9.7.8 设置“valve timing”子系统属性

延续上面小节的步骤。

step 1 设置“valve timing”子系统的模块。双击上面系统中的“valve timing”模块，添加对应的系统模块，如图 9.91 所示。

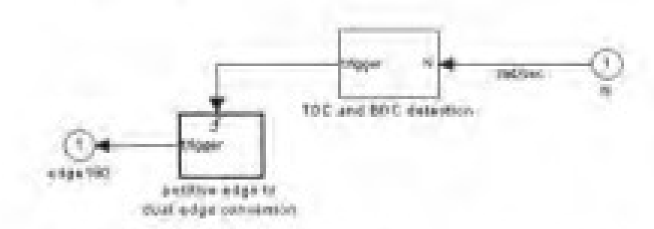


图 9.91 设置“valve timing”子系统的模块

在上面的程序模块中，首先通过输入端口输入发动机速度进行 TDC（上死点）和 BDC（下死点）的检测，输出触发的信号，然后设置触发器模块的属性，得出输出信号。

step 2 添加 "TDC and BDC detection" 子系统模块。双击上面系统中的 "TDC and BDC detection" 模块，打开对应的添加模块，在其中添加系统模块，如图 9.92 所示。

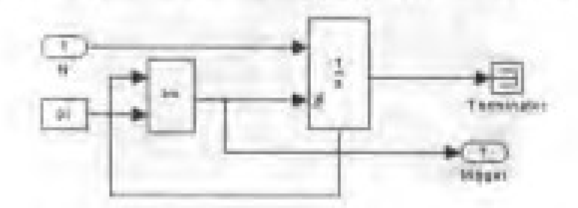


图 9.92 "TDC and BDC detection" 模块

在上面的模块中，将发动机的转速进行积分得到发动机旋转的弧度，然后将弧度和进行比较，如果积分的结果超过 π，就将积分器进行重设，重新运行积分。

step 3 设置积分器模块的属性。双击上面系统中的积分器模块，打开对应的模块属性对话框，设置积分器属性如图 9.93 所示。



图 9.93 设置积分器的属性

step 4 设置 "positive edge to dual edge conversion" 子系统模块。双击上面系统中的 "positive edge to dual edge conversion" 模块，在其中添加系统模块，如图 9.94 所示。

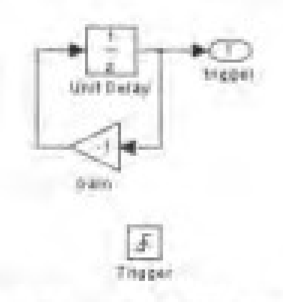


图 9.94 设置子系统模块

9.7.9 运行仿真系统

延续上面小节的步骤。

step 1 运行仿真系统。前面的步骤已经完成整个系统的模块设置，现在可以运行整个系统，得到仿真结果，如图 9.95 所示。



图 9.95 发动机速度

step 2 分析仿真结果。从上面的结果中可以看出，发动机的速度有一个启动过程，然后在 10s 左右保持稳定，达到最后的均衡速度。

9.8 S 函数（S-Function）

在 Simulink 中，S 函数（S-Function）或者系统函数（System Function）是用户借以自建 Simulink 模块所需要的，而且具有调用格式的函数文件。实质上，S 函数是一种描述动态系统的计算机语言，既可以直接使用 M 函数文件编写，也可以使用 C、C++、Ada 或者 FORTRAN 语言等语言编写。通过 C、C++、Ada 或者 FORTRAN 语言编写的 S 函数将需要通过编译而生成 MEX 文件，然后就可以像其他 MEX 文件一样动态连接 MATLAB。限于篇幅，本节只介绍使用 MATLAB 语言来编写的 S 函数。

9.8.1 S 函数概述

在 Simulink 中，S 函数采用一种特殊的调用语法使得函数可以和 Simulink 方程解法器进行交互，这种形式的交互和解法器和 Simulink 系统自身提供的模块之间的交互十分相似。S 函数的形式比较通用，可以使用 S 函数来描述连续、离散和混合系统。

一般而言，S 函数可以使用在下面的场合：

- 生成用户自行研究中可能反复调用的 S 函数模块；
- 可以创建代表硬件驱动的模块；
- 可以通过 S 函数将某个系统描述成一组数学方程组；
- 构建用于图形动画表现的 S 函数模块。

使用 S 函数的最主要的优点就是 S 函数模块可以被重用于各种场合，该 S 函数模块又可以通过设置不同的参数来显示不同的特性。



在 Simulink 中，只需要遵循一些简单的原则，就可以将自己设计的算法用 S 函数来实现。当编写好自己的 S 函数后，就可以在后续操作中调用该函数，并封装该函数。

9.8.2 S 函数的运行机理

为了帮助用户更好地理解 S 函数，在详细介绍 S 函数的各种属性之前，首先简要地介绍 S 函数的基本运行机理。在 Simulink 仿真过程中，将各种状态方程对应成为不同的仿真阶段，在仿真的开始

和结束，还包括系统的初始化和结束任务两个阶段。在上面的每一个仿真阶段，Simulink 都会重复地调用模型。

Simulink 的仿真流程如图 9.96 所示。

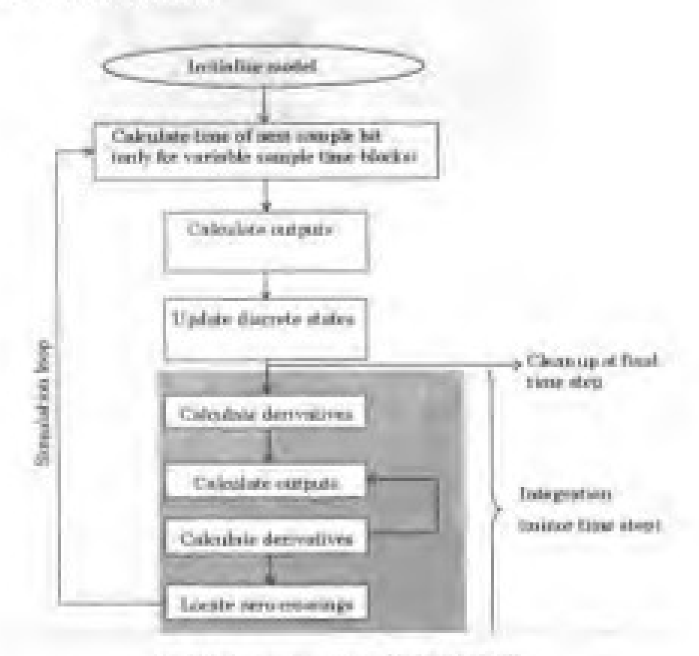


图 9.96　Simulink 的仿真流程

根据上面的流程图，在仿真的初始化阶段，Simulink 会将库中的模块并入用户自行创建的模型中，确定模块端口中的数据类型、采样时间，并评估模块的参数，决定模块运行的优先级，然后进行仿真循环，每一次的仿真循环就被称为仿真步。在每一个仿真步的不同仿真阶段，系统按照优先级运行模块。当系统完成某个仿真循环后，就会进入下一个循环仿真步，如此循环直到仿真结束。



说明

如果在仿真运行中，设置了积分的时间步长，那么 Simulink 会对时间步长进行细化，然后进行系统的仿真。

在 Simulink 中，S 函数是由一系列回调方法组成的。在每个仿真阶段，Simulink 会调用相应的回调方法来执行相应的任务，和一般模型的仿真类似，S 函数的回调方法会完成和流程类似的任务，这里就不重复介绍了。

9.8.3　S 函数模板

在 MATLAB 的早期版本中，S 函数的编写工作比较随意，对于开发人员和用户而言，这种随意的要求使得用户在 S 函数的编写上用的精力和时间都比较多，而在 MATLAB6.x 版本后，MATLAB 提供编写 S 函数的 M 文件标准模板，使得 S 函数的开发效率大大提高，开发的 S 函数结果可靠性也显著提高。

其中，S 函数 M 文件形式的标准模板是特殊的 M 文件，名为 sfuntmpl.m。默认情况下，其保存路径的目录为 MATLAB7.0\toolbox\simulink\blocks。为了简便描述该函数文件，下面简要给出该模板文件，并给出对应的说明文字。

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)
% 函数名称是模板对应的文件名称，在创建自行的 S 函数时，应该重新起名
% 关于该函数的输入变量名称、数目和次序，一般情况下不要改动
% 可以根据特殊的需要，在上面的输入变量基础上添加其他数目的变量
```

```
% 在上面的参数中，flag 是标记变量，共有 6 个不同的取值，分别代表 6 个不同的子函数

switch flag,
  case 0,
   [ sys,x0,str,ts] =mdlInitializeSizes;               % 调用初始化的子函数
  case 1,
    sys=mdlDerivatives(t,x,u);                      % 调用计算模块导数的子函数
  case 2,
    sys=mdlUpdate(t,x,u);                           % 调用更新模块离散状态的子函数
  case 3,
    sys=mdlOutputs(t,x,u);                          % 调用计算模块输出的子函数
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);              % 调用计算下一个采样时点的子函数
  case 9,
    sys=mdlTerminate(t,x,u);                        % 调用结束仿真的子函数
  otherwise
    error([ 'Unhandled flag = ',num2str(flag)] );
end
%==========================================================================%
function [ sys,x0,str,ts] =mdlInitializeSizes
sizes = simsizes;            % 调用 simsizes 函数，返回规范格式的 sizes 构架
                             % 这是一个通用的函数语句，用户不要轻易修改
sizes.NumContStates  = 0;    % 计算系统模块的连续状态数目，0 是默认数值
                             % 用户应该对自己创建的系统进行修改
sizes.NumDiscStates  = 0;    % 计算系统模块的离散状态数目，0 是默认数值
                             % 用户应该对自己创建的系统进行修改
sizes.NumOutputs     = 0;    % 计算系统模块的输出数目，0 是默认数值
                             % 用户应该对自己创建的系统进行修改
sizes.NumInputs      = 0;    % 计算系统模块的输入数目，0 是默认数值
                             % 用户应该对自己创建的系统进行修改
sizes.DirFeedthrough = 1;    % 计算系统模块中直接通向返回路线的数目，1 是默认数值
                             % 用户应该对自己创建的系统进行修改
sizes.NumSampleTimes = 1;    % 计算系统模块中采样时间的数目，1 是默认数值
                             % 用户应该对自己创建的系统进行修改
sys = simsizes(sizes);       %初始化后的构架sizes经过simsizes函数运算后向sys赋值
                             % 这是系统默认的命令，用户不要轻易修改
x0  = [];                    % 向模块的初始值赋值，其中[ ] 是默认数值
                             % 用户应该对自己创建的系统进行修改
str = [];
ts  = [0 0];
%==========================================================================%
function sys=mdlDerivatives(t,x,u)
% 编写计算导数向量的命令
sys = [];
%==========================================================================%
function sys=mdlUpdate(t,x,u)
% 编写计算更新模块离散状态的命令
sys = [];
%==========================================================================%
function sys=mdlOutputs(t,x,u)
% 编写计算模块输出向量的命令
sys = [];
%==========================================================================%
function sys=mdlGetTimeOfNextVarHit(t,x,u)
% 该函数只有在"变采样时间"条件下使用
sampleTime = 1;
```

```
sys = t + sampleTime;
%===================================================================%
function sys=mdlTerminate(t,x,u)
sys = [];
```

在上面的程序代码模板中，多次引用系统函数simsizes；默认情况下，其保存路径的目录为MATLAB7.0\toolbox\simulink\simulink。该函数的主要目的在于设置某个S函数的大小，具体的函数程序代码如下：

```
function sys=simsizes(sizesStruct)
switch nargin,
  case 0,  % return a sizes structure
    sys.NumContStates  = 0;
    sys.NumDiscStates  = 0;
    sys.NumOutputs     = 0;
    sys.NumInputs      = 0;
    sys.DirFeedthrough = 0;
    sys.NumSampleTimes = 0;
  case 1,  % convert a sizes structure into an array, or the other way around
     if ~isstruct(sizesStruct),
      sys = sizesStruct;
      if length(sys) < 6,
        error('Length of sizes array must be at least 6');
      end
      clear sizesStruct;
      sizesStruct.NumContStates  = sys(1);
      sizesStruct.NumDiscStates  = sys(2);
      sizesStruct.NumOutputs     = sys(3);
      sizesStruct.NumInputs      = sys(4);
      sizesStruct.DirFeedthrough = sys(6);
      if length(sys) > 6,
        sizesStruct.NumSampleTimes = sys(7);
      else
        sizesStruct.NumSampleTimes = 0;
      end
    else,
      % validate the sizes structure
      sizesFields=fieldnames(sizesStruct);
      for i=1:length(sizesFields),
        switch (sizesFields{i})
          case { 'NumContStates', 'NumDiscStates', 'NumOutputs',...
                 'NumInputs', 'DirFeedthrough', 'NumSampleTimes' },
          otherwise,
            error(['Invalid field name ''', sizesFields{i}, '''']);
        end
      end
      sys = [ ...
        sizesStruct.NumContStates,...
        sizesStruct.NumDiscStates,...
        sizesStruct.NumOutputs,...
        sizesStruct.NumInputs,...
        0,...
        sizesStruct.DirFeedthrough,...
        sizesStruct.NumSampleTimes ...
      ];
```

```
    end
end
```



在这里之所以详细列出系统函数simsizes的代码，是为了帮助读者更好地理解上面的函数模板代码，在这里就不详细展开分析了。

下面对模板函数代码进行简要的说明。

在函数 function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)中，对应的输入参数如下：

- t 表示当前时刻，采用绝对计量的时间数值；
- x 表示模块的状态向量；
- u 表示模块的输入向量；
- flag 程序的标记变量，对应不同的操作类型。

其对应的状态过程模块如图 9.97 所示。



图 9.97　系统的状态过程模块

9.8.4　添加 S 函数模块

在本小节中，将用一个简单的仿真系统演示如何在 Simulink 中使用 S 函数编写程序模块，达到有限积分的功能，下面分步骤详细介绍。

例 9.10　创建一个简单的实例，说明如何创建 S 函数。

step 1 打开 Simulink 的空白模板编辑器，向其中添加正弦函数模块，将其振幅设置为 5，该函数模块将作为输入信号，如图 9.98 所示。



图 9.98　添加输入信号模块

step 2 添加S函数模块和示波器模块。选择Simulink中"User-Defined Functions"模块库中的S-Function模块，将其添加到模块编辑器中，然后将其名称改为"Limit integrator by m-file S-Function"，如图9.99所示。



图9.99　添加S函数模块

step 3 封装S函数模块。选择上面模块中的S函数模块，单击鼠标右键，在弹出的快捷菜单中选择"Mask S-function"选项，如图9.100所示。

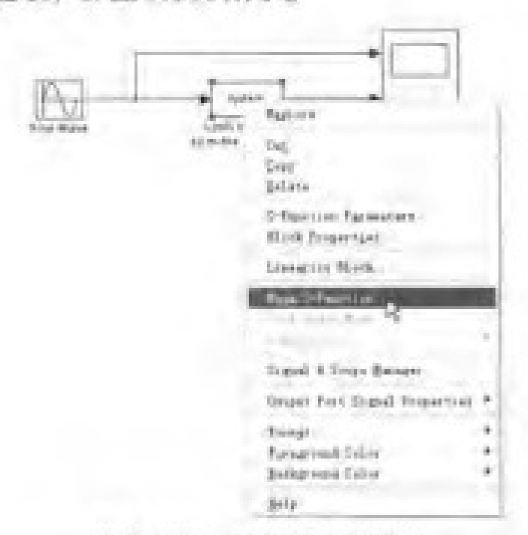


图9.100　封装S函数模块

在选择对应的菜单选项后，就会弹出对应的封装对话框，选择"Icon"选项卡，在其中设置封装子系统的图标，如图9.101所示。



图9.101　设置封装子系统的图标

step 4 设置仿真模块的参数。选择上面对话框中的"Parameters"选项卡，在其中设置仿真模块的参数值，如图9.102所示。



图 9.102 设置仿真模块的参数



在上面的对话框中，分别设置了积分的上限、下限和初始状态，其中变量名称分别和 S 函数中对应的名称相对应。

step 5 设置仿真系统的说明文字。选择上面对话框中的 "Documentation" 选项卡，在其中设置封装子系统的说明文字，如图 9.103 所示。



图 9.103 设置仿真系统的说明文字

step 6 查看封装后的子系统模块。完成所有设置后，系统模块如图 9.104 所示。



图 9.104 完成后的系统模块

9.8.5 添加 S 函数程序代码

延续上面小节的步骤。

step 1 设置仿真系统的函数参数。选择上面模块中的 S 函数模块，单击鼠标右键，在弹出的快捷菜单中选择 "S-function Parameters" 选项，如图 9.105 所示。

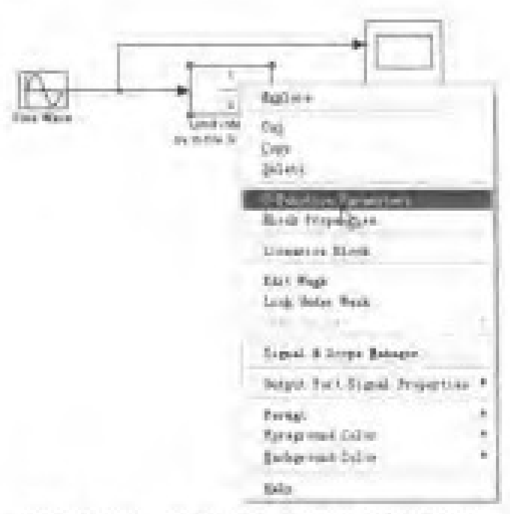


图 9.105　设置仿真系统的函数参数

当选择对应的菜单选项后，可以打开 S 函数参数对话框，如图 9.106 所示。

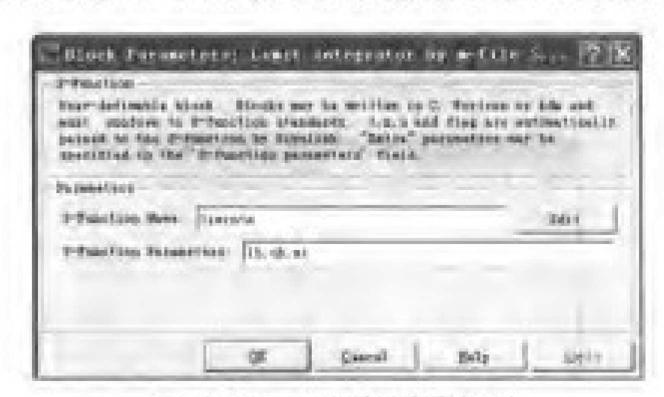


图 9.106　S 函数的参数设置

step 2 添加S函数的程序代码。在对话框中的"S-Function Name"选框中输入函数名称"limintm"，在"S-Function Paremeters"选框中输入参数名称"lb,ub,xi"。单击上面对话框中的"Edit"按钮，就可以查看对应的 S 函数，其代码如下：

```
function [ sys,x0,str,ts] =limintm(t,x,u,flag,lb,ub,xi)
%计算表达式的有限积分数值
%判断参数 flag 的数值
switch flag
  case 0
    [ sys,x0,str,ts]  = mdlInitializeSizes(lb,ub,xi);
                        %调用 mdlInitializeSizes 函数
  case 1
    sys = mdlDerivatives(t,x,u,lb,ub);    %调用 mdlDerivatives 函数
  case {2,9}
    sys = [];       %不进行任何的操作
  case 3
    sys = mdlOutputs(t,x,u);    %调用 mdlOutputs 函数
  otherwise
    error(['unhandled flag = ',num2str(flag)]);    %显示错误信息
end
%=============================================================================%
function [ sys,x0,str,ts]  = mdlInitializeSizes(lb,ub,xi)
sizes = simsizes;
```

```
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
str = [];
x0  = xi;
ts  = [0 0];    % 样本时间
%==========================================================================
function sys = mdlDerivatives(t,x,u,lb,ub)
if (x <= lb & u < 0)  | (x>= ub & u>0 )
  sys = 0;
else
  sys = u;
end
%=========================================================================%
function sys = mdlOutputs(t,x,u)
sys = x;  % 返回输入参数 x 的数值
```



上面的程序代码就是对应的 S 函数，其整体的框架和 Simulink 提供的 S 函数模板大致相同，这里就不重复介绍了。

step 3 设置仿真运行的参数。双击系统的 S 函数模块，打开对应的模块参数对话框，在其中设置积分参数，如图 9.107 所示。

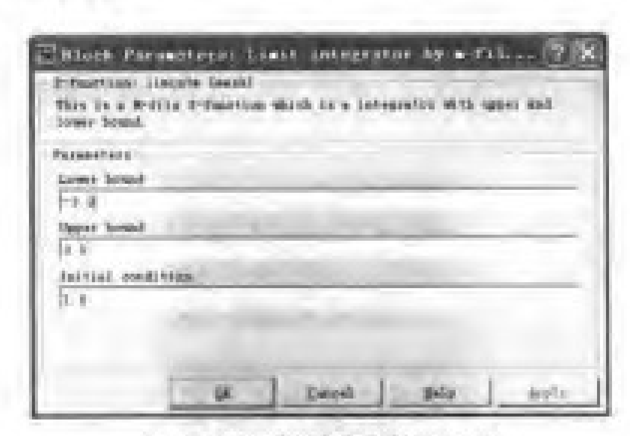


图 9.107　设置积分的上下限

在上面的对话框中，将积分的上限设置为 3，积分的下限设置为 -3，然后将积分的初始值设置为 1，单击 "Apply" 按钮，保存上面所有的设置情况。最后单击 "OK" 按钮，关闭上面的对话框。

9.8.6　运行仿真

延续上面小节的步骤。

step 1 运行系统仿真。将系统仿真的时间设置为 14，然后单击 "开始仿真" 按钮，得到的仿真结果如图 9.108 所示。



图 9.108　得到仿真结果

step 2 修改系统仿真参数，重新运行仿真。双击 S 函数模块，打开函数参数的设置对话框，在其中重新设置系统参数，如图 9.109 所示。

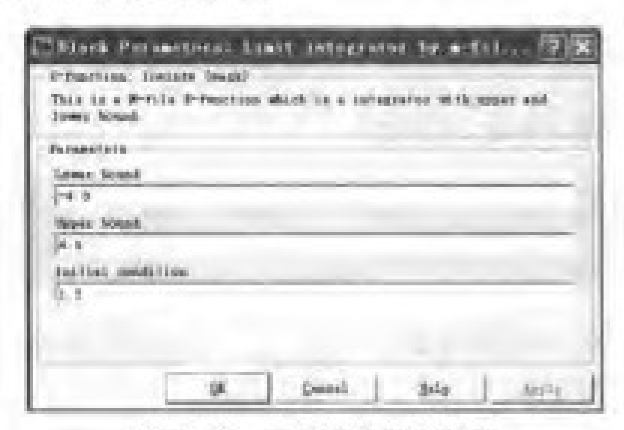


图 9.109　重新设置仿真参数

step 3 重新运行仿真。设置上面的参数后，单击 "OK" 按钮，然后重新运行仿真，结果如图 9.110 所示。

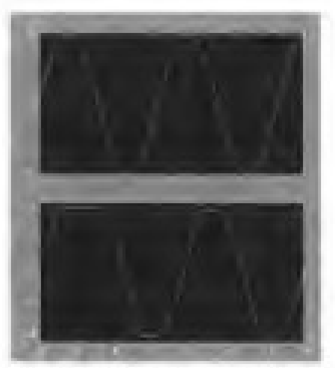


图 9.110　新的仿真结果

9.9 S 函数实例

本节将介绍如何在 Simulink 中使用 S 函数来模拟动态物体和弹簧系统的运动，首先根据物理知识列出物体的状态微分方程，然后根据微分结果变量参数编写对应的 S 函数，实现实体动画的演示。同时，本实例还将涉及到读取数据、状态分析等关于 Simulink 的多种知识。

9.9.1 添加系统模块

例 9.11 创建两个物体和弹簧系统振荡运动的仿真模型。

step 1 添加系统模块。由于本实例中的系统模块比较复杂，首先给出所有的系统模块连接方法，然后分步骤详细介绍每个模块的属性，如图 9.111 所示。

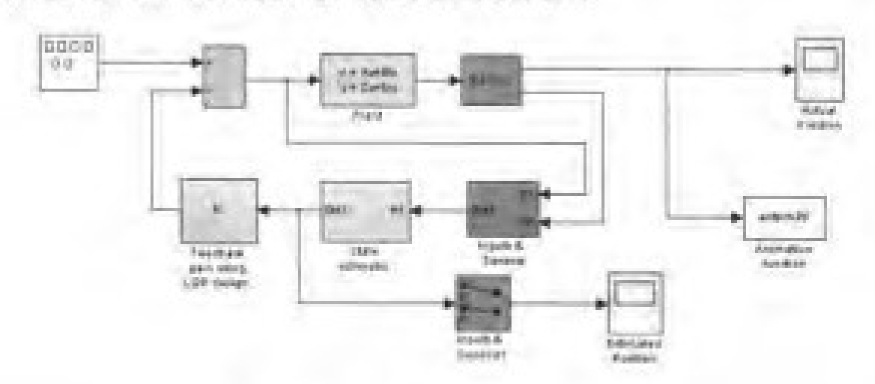


图 9.111 添加系统模块

step 2 设置输入信号的属性。在本实例中，输入信号是方波，代表左侧物体所承受的周期外力，其具体的属性如图 9.112 所示。

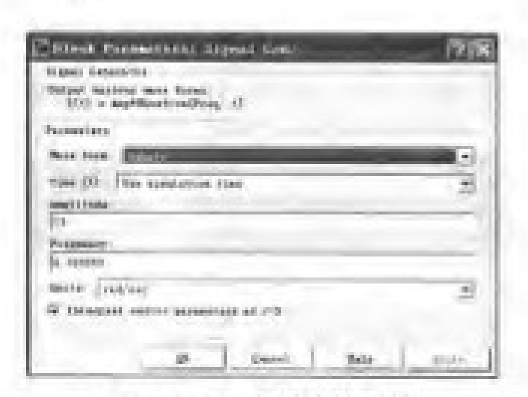


图 9.112 方形波的属性

step 3 设置 "Plant" 模块的属性。在本实例中，"Plant" 模块属于 "State Space" 模块库，代表该系统运动的微分方程组，其属性如图 9.113 所示。



图 9.113 设置 "Plant" 模块的属性

step 4 分析 "Plant" 模块的原理。该模块定义的微分方程组是系统运动的核心方程组，因此有必

要在这里详细展开介绍。根据上面模块参数，对应的微分方程组如下：

$$
\begin{cases}
\dfrac{\mathrm{d}^2 x_1(t)}{\mathrm{d}t^2} = -\dfrac{Ka}{m_1}x_1(t) + \dfrac{Ka}{m_1}x_3(t) + u \\
\dfrac{\mathrm{d}x_1(t)}{\mathrm{d}t} = x_2(t) \\
\dfrac{\mathrm{d}^2 x_3(t)}{\mathrm{d}t^2} = \dfrac{Ka}{m_2}x_1(t) - \dfrac{Ka}{m_2}x_3(t) \\
\dfrac{\mathrm{d}x_3(t)}{\mathrm{d}t} = x_4(t)
\end{cases}
$$

在上面的方程组中，参量$x_1(t)$、$x_3(t)$分别表示左边、右边物体的位移（或者被认为坐标），m_1、m_2则分别表示左边和右边物体的质量，$x_2(t)$和$x_4(t)$是方程组中的中间变量，其物理含义分别是两个物体运动的速度，但是在本实例中并不处理这些变量，只是用作设置方程组的中间变量；u表示左侧物体所承受的外力（也就是外部信号），其由两个部分组成，其中一部分是方形波输入信号，另外一个部分则是自反馈的信号。

然后根据下一个方程 $\boldsymbol{y=Cx+Du}$，得到的关系等式如下：

$$
y = \begin{bmatrix} x_1(t) \\ x_3(t) \\ x_3(t) \end{bmatrix}
$$

其中 y 表示的是该模块的输出信号，参量 $x_1(t)$、$x_3(t)$分别表示左边、右边物体的位移。



从上面的分析可以看出，该 "State Space" 模块输入物体系统的受力信号，然后根据运动的微分方程组，得到两个物体运动的位移信号。

step 5 设置 "Dmux" 模块的属性。在本实例中，"Dmux" 模块的功能在于将上面模块中的输出信号进行分割，其属性如图 9.114 所示。

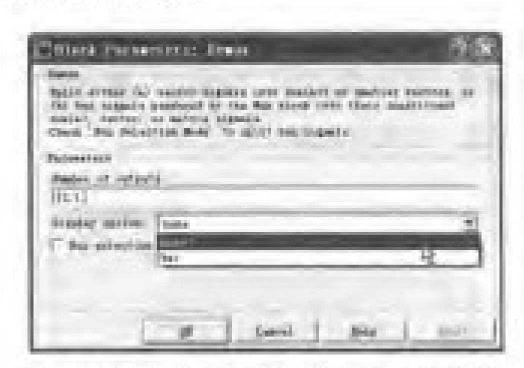


图 9.114　设置 "Dmux" 模块的属性

在上面的对话框中，将 "Display option" 选项设置为 "none"，同时取消选中 "Bus selection" 选项，因此该模块在外观上就和普通模块一致。同时，将上个步骤的输出信号分割为一个二维向量y_1和一个一维向量信号 y_2。其中，二维向量信号 y_1 为：

$$
y_1 = \begin{bmatrix} x_1(t) \\ x_3(t) \end{bmatrix}
$$

因此，和"Actual Position"示波器模块相连的输入信号 y_1 是两个物体运动的位置，在进行实体仿真过程中，该示波器将显示两个物体的位移情况。

9.9.2 添加 S 函数的程序代码

延续上面小节的步骤。

step 1 设置"Animation function"模块的属性。在本实例中，"Animation function"模块的功能在于设置 S 函数实现实物的仿真，其属性如图 9.115 所示。

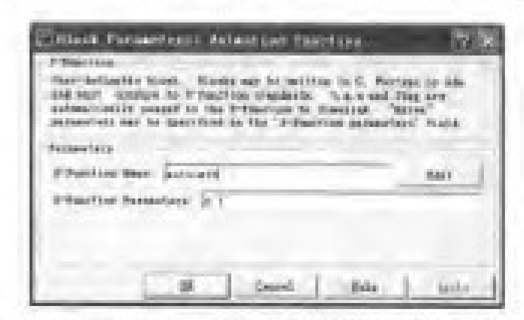


图 9.115 设置"Animation function"模块的属性

step 2 添加 S 函数的程序代码。在上面的对话框中，单击"Edit"按钮，就可以打开仿真函数的代码：

```
function [ sys,x0]=antsim38(t,x,u,flag,ts);
% 定义全局变量
global xSpr2 xBx12 xBx22 sim38
% 定义样本时间的偏移数值
offset=4;
if flag==2,
  if any(get(0,'Children')==sim38),
    if strcmp(get(sim38,'Name'),'sim38 Animation'),
      set(0,'currentfigure',sim38)
      u(2)=u(2)+offset;
      distance=u(2)-u(1);
      hndl=get(gca,'UserData');
      x=[ xBx12+u(1); xSpr2/4*distance+u(1); xBx22+distance+u(1)];
      set(hndl,'XData',x);
      drawnow;
    end
  end
  sys=[ ];
elseif flag == 4 % 返回下一个样本
    % ns 表示样本的数据
  ns = t/ts;
  % 返回时间变量数值
  sys = (1 + floor(ns + 1e-13*(1+ns)))*ts;
elseif flag==0,
  %初始化仿真系统的图形
  animinit('sim38 Animation');
  sim38 = findobj('Type','figure','Name','sim38 Animation');
  axis([ -10 20 -7 7]);
  hold on;
```

```
%定义变量的数值
xySpr2=[  ...
        0.0       0.0
        0.4       0.0
        0.8       0.65
        1.6      -0.65
        2.4       0.65
        3.2      -0.65
        3.6       0.0
        4.0       0.0];
  xyBx12=[  ...
        0.0       1.1
        0.0      -1.1
       -2.0      -1.1
       -2.0       1.1
        0.0       1.1];
  xyBx22=[  ...
        0.0       1.1
        2.0       1.1
        2.0      -1.1
        0.0      -1.1
        0.0       1.1];
 %定义运动物体的坐标
   xBx12=xyBx12(:,1);
   yBx12=xyBx12(:,2);
   xBx22=xyBx22(:,1);
   yBx22=xyBx22(:,2);
   xSpr2=xySpr2(:,1);
   ySpr2=xySpr2(:,2);
   x=[ xBx12; xSpr2; xBx22(:,1)+offset];
   y=[ yBx12; ySpr2; yBx22];
   % 绘制两个滑动物体下的平板
   plot([-10 20],[-1.3 -1.3],'yellow', ...
        [-10:19;-9:20],[-2 -1.3],'yellow','LineWidth',2);
   hndl=plot(x,y,'y','EraseMode','background','LineWidth',3);
   set(gca,'UserData',hndl);
   sys=[ 0 0 0 2 0 0];
   x0=[ ];
 end;
```

在上面的代码中，首先创建了新的图形界面，然后在该图形窗口界面上添加物体对象，并通过高级句柄语言来设置物体运动的坐标信息，实现最后的动态运动。



关于上面代码中引用的多种变量，其具体的数据来自关于该系统的运动参数数据文件，关于该数据文件的加载工作将在后面介绍。

9.9.3 添加子系统模块

延续上面小节的步骤。

step 1 添加"Inputs & Sensors"子系统的模块。双击"Inputs & Sensors"模块，打开模块编辑器，在其中添加子系统的模块，如图 9.116 所示。

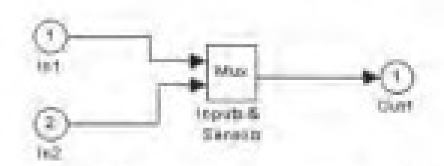


图 9.116　添加 "Inputs & Sensors" 子系统的模块

在本实例中 "Inputs & Sensors" 子系统的功能是将两组信号合并为一个向量，其中一组信号为 u（也就是上面的合成的输入信号），另外一组信号是一维向量信号 $y_2=x_1(t)$，得到的输出结果信号是：

$$y=\begin{bmatrix} u \\ x_1(t) \end{bmatrix}$$

step 2 添加 "State estimator" 子系统的模块。双击 "State estimator" 模块，打开模块编辑器，在其中添加子系统的模块，如图 9.117 所示。

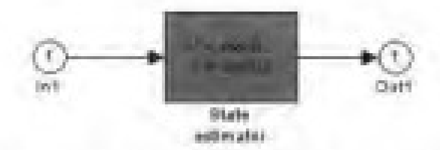


图 9.117　添加 "State estimator" 子系统的模块

step 3 设置 "State estimator" 模块的属性。在上面的子系统中，"State estimator" 模块的属性对话框如图 9.118 所示。



图 9.118　设置模块的属性

step 4 查看参数值。在上面的参数对话框中，ae、be、ce 和 de 都是在运行仿真系统之前加载的变量参数，其具体的数值可以使用 MATLAB 的相关命令获得，结果如下：

```
>> ae
ae =
         0    1.0000  -31.4507         0
   -1.0000         0   44.8537         0
         0         0  -46.6599    1.0000
    1.0000         0  -89.5750         0
>> be
be =
```

```
         0   31.4507
    1.0000  -43.8537
         0   46.6599
         0   88.5750
>> ce
ce =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
>> de
de =
     0     0
     0     0
     0     0
     0     0
```

从上面的结果中可以看出，ae和ce是一个4×4的矩阵，be和de是一个2×4矩阵，根据"State Space"模块的属性特点，该模块的输出变量是一个四维的向量，其次序和前面步骤中设置的变量次序相同。



关于该微分方程组的含义，读者可以参看前面的步骤进行分析，和前面的步骤一个明显的不同在于，其输入变量是一个二维向量，而不是一维标量。

step 5 设置"Inputs & Sensors1"模块的属性。在本系统中，"Inputs & Sensors1"模块的功能是进行信号选择，其属性如图9.119所示。



图9.119　设置信号选择的属性

在上面的对话框中，在"Elements"选框中输入"[1 3]"，表示选择输入信号的第一列和第三列的信号，分别表示通过模拟系统微分方程得到的两个物体运动的位移；在"Input port width"选框中输入"4"，表示输入信号为4维向量，也就是通过"State estimator"子系统的模块输出的$x_1(t)$、$x_2(t)$、$x_3(t)$和$x_4(t)$信号。

step 6 添加"Feedback gain using LQR design"子系统的模块。在本实例中，该模块的功能是输出反馈信号，其具体的子系统模块如图9.120所示。

图 9.120　添加反馈信号的模块

step 7 设置 "Feedback gain using LQR design" 模块的属性。双击 "Feedback gain using LQR design" 模块，打开对应的属性对话框，在其中设置模块参数，如图 9.121 所示。

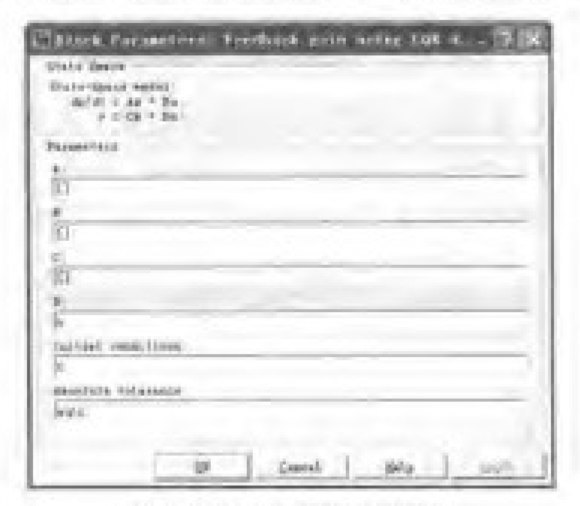


图 9.121　设置模块的属性



在上面的模块对话框中的参数 k 也是在运行系统之前加载的数值变量，具体数值可以在 MATLAB 命令窗口中查看相应的数值，这里就不重复介绍了。

step 8 封装上面的子系统。由于上面子系统的功能是对参数进行放大，所以有必要将其进行封装，设置封装系统图标的对话框如图 9.122 所示。



图 9.122　设置封装子系统的图标

设置封装子系统的参数对话框如图 9.123 所示。

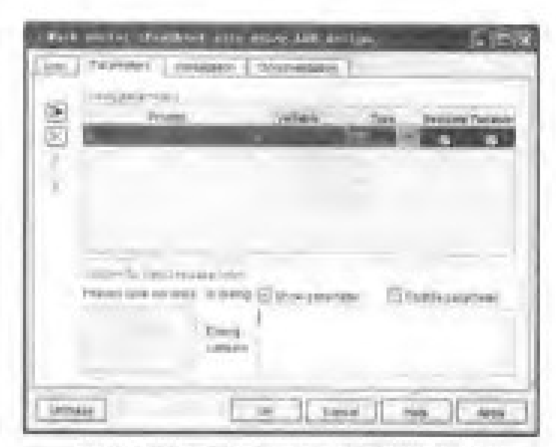


图 9.123　设置封装子系统的参数

step 9 设置系统的模块属性。在模块编辑器中单击鼠标右键，在弹出的快捷菜单中选择 "Model Properties" 选项，打开 "Model Properties" 对话框，选择 "callbacks" 选项卡，在 "Model pre-load fucntion" 选框中输入 "load sim38"，如图 9.124 所示。

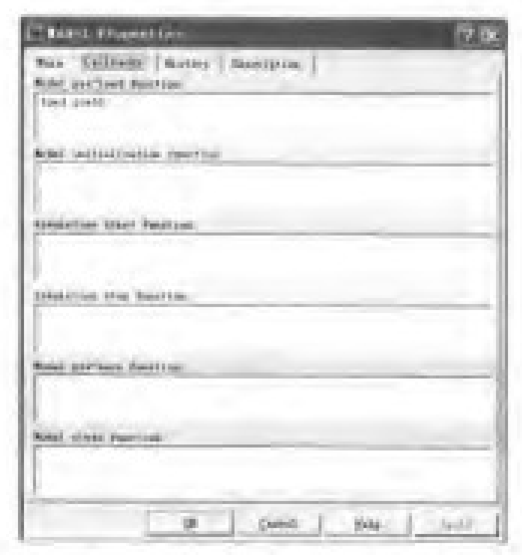


图 9.124　设置系统的加载函数

step 10 查看仿真系统的数据文件。在上面的命令行中，sim38 是关于该系统参数的 mat 数据文件，可以通过加载数据的命令来查看该数据文件的内容，如图 9.125 所示。

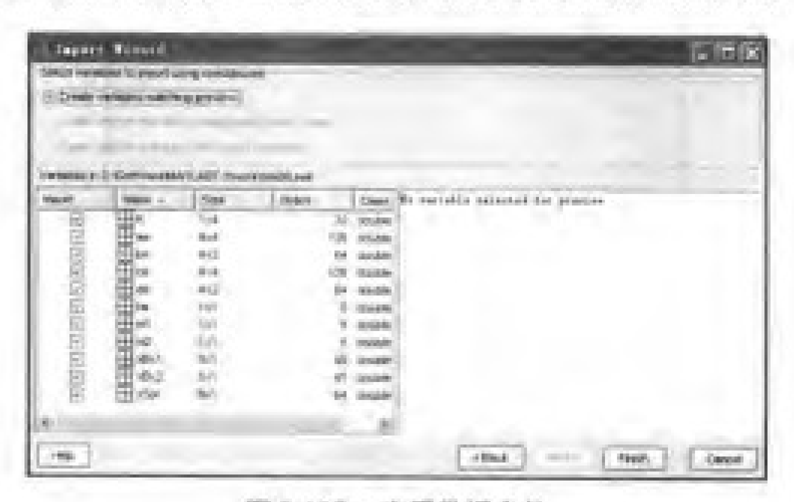


图 9.125　查看数据文件

9.9.4 运行仿真系统

延续上面小节的步骤。

step 1 运行系统仿真。将系统的仿真时间设置为 10 000，然后单击"开始仿真"按钮，得到动态仿真的结果，选取其中两个仿真画面来显示仿真结果，如图 9.126 和图 9.127 所示。



图 9.126 动态仿真画面 1

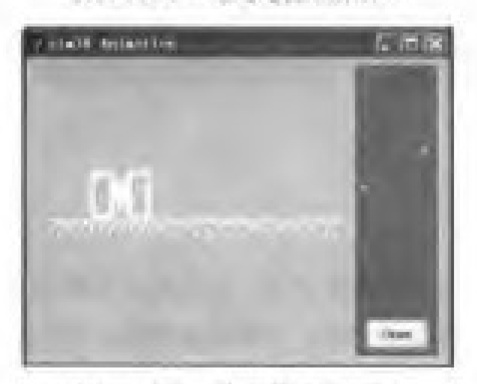


图 9.127 动态仿真画面 2

step 2 查看示波器的图形。用户可以查看两个物体位置的变化情况，其中"Actual Position"示波器中显示的图表如图 9.128 所示。

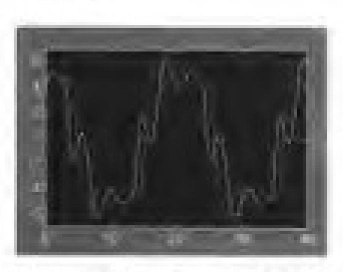


图 9.128 两个物体的真实位移图形

而通过系统仿真得到的模拟位移图形如图 9.129 所示。

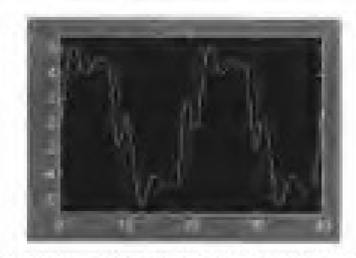


图 9.129 通过仿真得到的位移图形

step 3 查看系统模块的变化。当仿真结束后，系统模块的改变情况如图 9.130 所示。

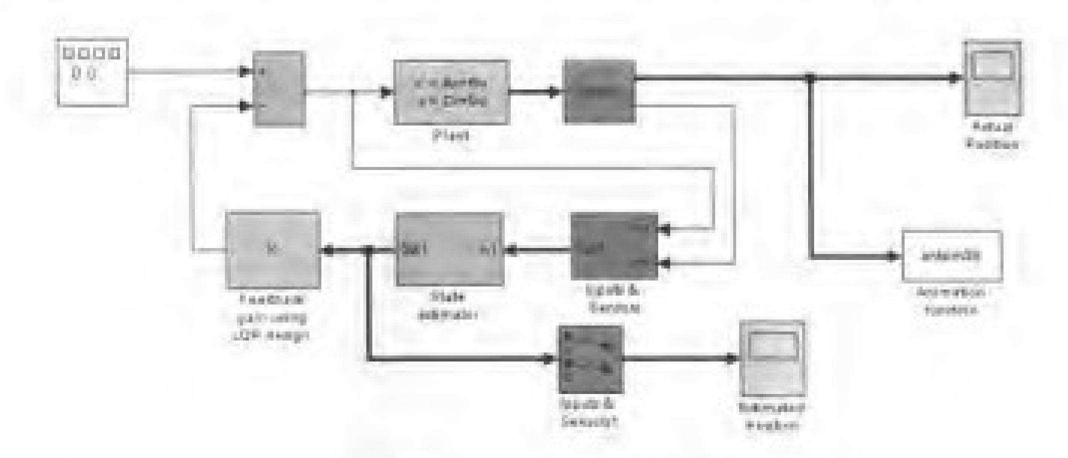


图 9.130　运行后的系统模块



由于在上面的模块中，有些模块的输入或者输出信号是多维向量，因此在运行系统仿真后，对应的信号连接线会被加粗，提示用户该信号是多维向量。

9.10 仿真结果分析

前面介绍了关于Simulink中几个创建的内容，本节将介绍如何对各种仿真系统的结果进行分析。分析仿真结果也是创建仿真系统的重要部分，了解仿真结果，对仿真结果进行处理，是仿真模型分析的重要步骤。

9.10.1 分析 Simulink 模型的特征

在某些情况下，需要在命令窗口中对Simulink中的模型属性进行分析，然后再根据仿真系统的属性特征进行处理。主要的属性包括：系统模型中包含多少连续状态、包含多少离散状态、模型中哪些模块对应着状态向量中的哪个分量等。

在前面介绍过，在本质上Simulink模型都是图形化的微分或者差分方程。无论高阶微分或者差分方程，还是传递函数，Simulink都用连续或者离散的状态方程加以描述。Simulink模型库中的积分器模块、传递函数模块和状态空间模块，都分别对应着连续或者离散的状态变量。

在Simulink中，从模型中获取状态信息的命令就是模型名称本身，但是不能包含扩展名，具体的调用格式如下：

```
>> [sizes,x0,StateCell]=model
```

下面以具体的实例来介绍该命令的使用方法。

例 9.12　使用命令来分析例9.5中的Anti-Lock Brake System的模块属性。

step 1 根据前面内容，该系统的模块如图 9.131 所示。

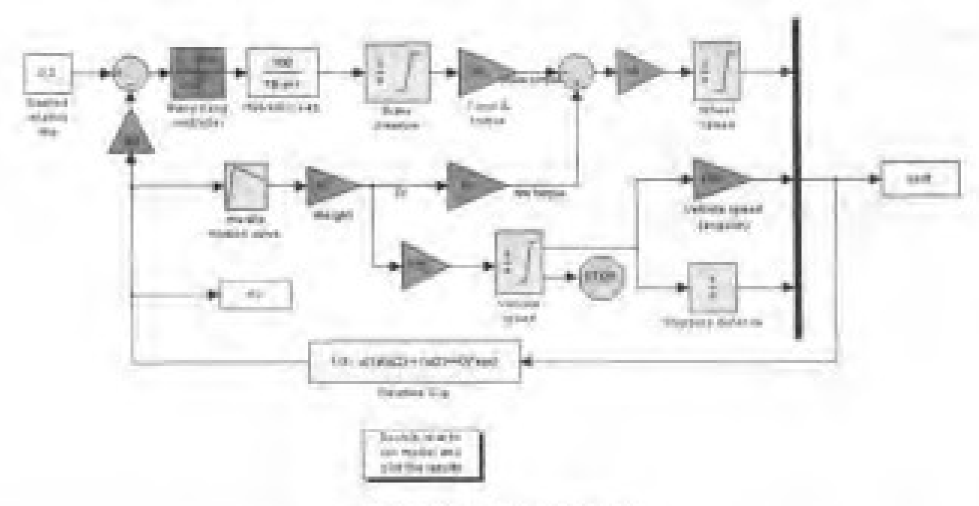


图 9.131 系统的模块

step 2 查看该模块的属性特征。在 MATLAB 的命令窗口中输入下面的命令

```
>> [ sizes,x0,StateCell] =sim32;
>> Sizes=sizes';X0=x0';
>> Sizes
Sizes =
      5     0     0     0     0     0     2
>> X0
X0 =
   88.0000   70.4000         0         0         0
>> StateCell
StateCell =
     'sim32/Vehiclespeed'
     'sim32/WheelSpeed'
     'sim32/Stopping distance'
     'sim32/Brakepressure'
     'sim32/Hydraulic Lag '
```



由于上面的系统中运行系统的仿真之前，需要加载关于系统参数的数据文件，因此在了解这些属性之前，需要首先运行仿真，加载对应的数据。

下面简要介绍上面代码的结果。

- Sizes：Sizes 是一个 7 维向量，对应各分量向量的含义如下。
  - Sizes(1)：表示状态向量中的连续分量个数。
  - Sizes(2)：表示状态向量中的离散分量个数。
  - Sizes(3)：表示输出分量个数。
  - Sizes(4)：表示输入分量个数。
  - Sizes(5)：表示系统中不连续解的个数。
  - Sizes(6)：表示系统中是否包含直通回路。
  - Sizes(7)：表示状态中不同采样速率的个数。
- X0：返回模型状态向量的初始数值。在 Simulink 中，积分器所设置的初始数值可以被模型窗口仿真参数对话框中的参数设置来修改，而在仿真参数对话框中设置的初始值又可以在命令窗口中的 X0 初始值来重新设置。

- StateCell：是一个元胞数组，依次给出所有状态变量对应模块所在的模块名称、子系统名称和模块名称。



根据上面的介绍，sim32 系统中包含5个连续状态变量，同时包含2个不同的采样速率，系统可以在时间上分成2个部分。

9.10.2 Sim 命令

当用户在模型窗口中直接运行某个系统的仿真时，每个运行都是针对某个固定的模块参数设置的，在运行过程中一般不能轻易修改这些参数的数值。MATLAB 中，允许使用命令来运行 Simulink 仿真。通过命令进行模型的仿真使得用户可以从M文件运行仿真，可以在程序代码中不断地修改模块参数，也可以让用户使用所有 M 文件中的循环结构实现复杂的仿真过程。

在 Simulink 中，使用 sim 命令运行 Simulink 仿真的具体调用格式如下：

- [t,x,y] = sim(model);
- [t,x,y] = sim(model,timespan,options,ut);
- [t,x,y1, y2, …, yn] = sim(model,timespan,options,ut);

关于上面调用格式中的参数说明如下。

- model：被运行的模型名称，不包含扩展名，但是，该模型文件必须在MATLAB的搜索路径上。
- y：输出矩阵，取自模型中输出端口模块的记录，y 中的第 k 列数值也就是第 k 个输出端口的时间变量记录。
- y1, y2, …, yn：每个向量都是列向量，分别输出 n 个输出端口模块上的记录。
- x：状态矩阵。每个数据列表示状态变量的记录。状态变量的排列次序可以从上面小节中的 StateCell 中获取。
- timespan：用来指定仿真的时间区间，可以选取下面的变量。
  - [ ]：空矩阵，表示使用模块编辑器中设置的仿真时间。
  - T_final：标量参数，指定系统仿真的终止时间。
  - [T_start T_final]：二维向量参数，指定系统仿真的起始和终止时间。
- OutputTimes：任何指定输出时间记录点的向量。
- Options：设置仿真参数中具有最高级别的优先权，它可以覆盖模型参数对话框中的参数设置。关于 Options 参数的结构和设置，将在后面的命令中详细介绍。
- ut：赋给仿真对象的输入端口模块的数值，具有最高的设置优先权。

9.10.3 Sim 命令实例

例 9.13 使用 sim 命令运行 Simulink 的仿真文件，动态设置关于该仿真系统的参数，然后运行得到相应的仿真结果。

step 1 查看原始的系统模块。在本例中选择的仿真文件是例 9.2 所创建的仿真系统 sim28 文件，原始的系统模块如图 9.132 所示。

图 9.132 原始的系统模块

step 2 返回到 MATLAB 的命令窗口中，输入下面的命令代码：

```
>> for i=1:5
op(i)=i;
Opts(i)=simset('InitialState',op(i));
[ t,x,y] = sim('Sim28',[ 0 10] ,Opts(i));
plot(t,x,'LineWidth',2),hold on
grid
end
```

在上面的代码中，通过使用循环结构为该线性系统设置不同的积分初始数值，然后通过sim命令在不同的初始条件下运行系统的仿真，最后，通过plot命令绘制通过sim命令获取的仿真数据。



在上面的步骤中，使用到simset命令，该命令的主要功能是设置编辑仿真参数，将在后面详细介绍。

step 3 查看上面代码的运行结果，在输入上面的代码后，按"Enter"键，得到的图形结果如图9.133所示。

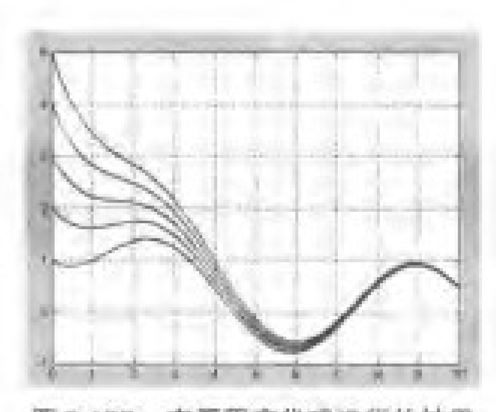


图 9.133 查看程序代码运行的结果

9.10.4 simset 命令

上面的例子提到过，使用simset命令可以设置编辑仿真参数。在Simulink中，simset命令的常见调用格式如下：

- options = simset(property, value, ...);
- options = simset(old_opstruct, property, value, ...);
- options = simset(old_opstruct, new_opstruct);
- simset

在上面的命令中，property参数的含义是指该系统某种参数属性，old_opstruct和new_opstruct分别代表设置系统的原来属性或者现在的属性数值。其中，可以使用simset显示关于设置参数的各种细节信息，得到的结果如下：

```
>> simset
           Solver: [ 'VariableStepDiscrete' |
                     'ode45' | 'ode23' | 'ode113' | 'ode15s' | 'ode23s' |
'ode23t' | 'ode23tb' | 'FixedStepDiscrete' |
                     'ode5' | 'ode4' | 'ode3' | 'ode2' | 'ode1' | 'ode14x' ]
           RelTol: [ positive scalar {1e-3} ]
           AbsTol: [ positive scalar {1e-6} ]
           Refine: [ positive integer {1} ]
          MaxStep: [ positive scalar {auto} ]
          MinStep: [ [positive scalar, nonnegative integer] {auto} ]
      InitialStep: [ positive scalar {auto} ]
         MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
        FixedStep: [ positive scalar {auto} ]
    ExtrapolationOrder: [ 1 | 2 | 3 | {4} ]
NumberNewtonIterations: [ positive integer {1} ]
     OutputPoints: [ {'specified'} | 'all' ]
  OutputVariables: [ {'txy'} | 'tx' | 'ty' | 'xy' | 't' | 'x' | 'y' ]
       SaveFormat: [ {'Array'} | 'Structure' | 'StructureWithTime']
    MaxDataPoints: [ non-negative integer {0} ]
       Decimation: [ positive integer {1} ]
     InitialState: [ vector {[]} ]
   FinalStateName: [ string {''} ]
             Trace: [ comma separated list of 'minstep', 'siminfo',
'compile', 'compilestats' {''}]
     SrcWorkspace: [ {'base'} | 'current' | 'parent' ]
     DstWorkspace: [ 'base' | {'current'} | 'parent' ]
        ZeroCross: [ {'on'} | 'off' ]
            Debug: [ 'on' | {'off'} ]
```



在上面的所有系统参数中，都可以使用simset命令来进行访问，但是在运行仿真时，只有一些常见的属性需要经常访问并编辑，其他参数将主要用于参考检测系统的性能。

9.10.5 simset命令实例

例9.14 使用simset设置仿真系统的不同求解器参数，然后比较两种不同参数得到的仿真结果。为了简便分析过程，将沿用上面的例子。

step 1 在MATLAB的命令窗口中输入下面的命令代码：

```
>> opts(1)=simset('Solver','ode15s');
[t,x,y] = sim('Sim28',[0 10],opts(1));
plot(t,x,'r','LineWidth',1.5)
hold on
opts(2)=simset('Solver','ode45');
[t,x,y] = sim('Sim28',[0 10],opts(2));
plot(t,x,'g','LineWidth',1.5)
axis([9 10 0.5 1])
grid
```

step 2 查看图形结果。在输入上面代码后，得到的图形如图9.134所示。

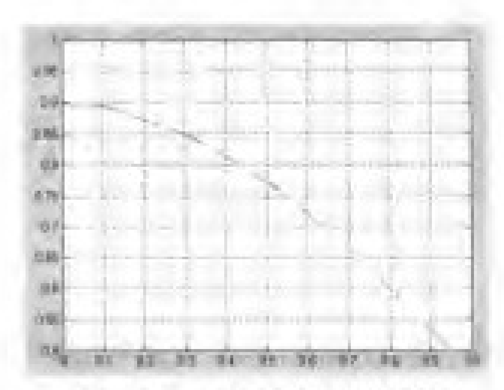


图9.134 运行程序得到的结果



在上面的程序结果中，使用了不同的解法器ode15s和ode45来求解同一个仿真系统，然后比较仿真结果中9~10s的图形。可以看出，使用不同的解法器会得出类似但是有差别的计算结果。

关于simset命令，笔者认为有必要提示读者下面的一些内容。

- 命令simset所设置的参数将在仿真过程中被优先使用，但是并不修改仿真系统对话框中的参数设置，对话框中的参数设置是持久的，而simset命令修改的参数值只限于使用在当前命令中。
- 关于系统模型的参数中包含SrcWorkspace和DstWorkspace两个比较特殊的变量，其中SrcWorkspace表示制定系统表达式中计算所在的空间，取值为字符串，分别为{'base'} | 'current' | 'parent'，依次表示MATLAB的基础工作空间、当前空间和调用当前系统的函数空间，其中current是默认数值；DstWorkspace则是指定"To Workspce"模块输出数据所在的空间，取值为字符串，分别为{'base'} | 'current' | 'parent'，含义相同。



simset命令获取的系统模块的参数只是一些高层属性，如果希望对系统的底层参数进行操作和编辑，则需要使用set_param命令来实现。感兴趣的读者可以查看相应的帮助文件。

在本小节的最后，将简要说明使用simget命令获取模型的信息。在Simulink中，simget命令的常用调用格式如下。

- struct = simget(model)
- value = simget(model, property)
- value = simget(OptionStructure, property)



上面命令中各参数的含义和前面各命令的参数含义相同，这里就不重复介绍了。其中property是使用simset可以访问的任何一个参数名称。

9.10.6 模型的线性化命令

如果希望利用发展成熟的线性分析方法来解决现实中的非线性问题，则需要研究如何将这些非线性问题进行线性化，然后研究线性化后的模型问题。

所谓的模型线性化是指将用户所建模型用状态空间矩阵A、B、C和D表示的线性方程组，状态空间矩阵按照下面的方程组的形式来描述线性的输入/输出关系：

$$\begin{cases} x' = Ax + Bu \\ y = Cx + Du \end{cases}$$

其中，x、u和y分别表示状态、输入和输出向量，系统中的输入/输出变量必须使用simulink提供的输入模块和输出模块。这是一种简化方程，只有在输入信号附近、输入信号时间点附近很小的范围内才成立。同时，该简化的线性方程并不是对所有的非线性系统都成立，有些非线性系统这种处理也会影响整个系统的性质。

在Simulink中，提供两种线性分析的命令，分别对连续系统和离散系统进行线性操作，其中对连续系统进行线性操作的命令如下：

```
argout = linmod('sys');
argout = linmod('sys',x,u);
argout = linmod('sys', x, u, para);
argout = linmod('sys', x, u, 'v5', para);
argout = linmod('sys', x, u, 'v5', para, xpert, upert);
```

通过使用上面的线性命令，不仅可以获取非线性系统的近似线性数学模型，也可以用来获取线性模块的数学描述。由于这样获取的数学模型中没有进行"零－极点"对消，有可能阶次比较高，可以使用Control Toolbox中的minreal命令来求取最小值。

9.10.7 模型的线性化实例

例9.15 演示如何使用Simulink来对非线性系统进行线性化。

step 1 添加非线性系统的模块。在本实例中，由于只是为了演示线性化的操作方法，因此添加的非线性模块比较简单，如图9.135 所示。

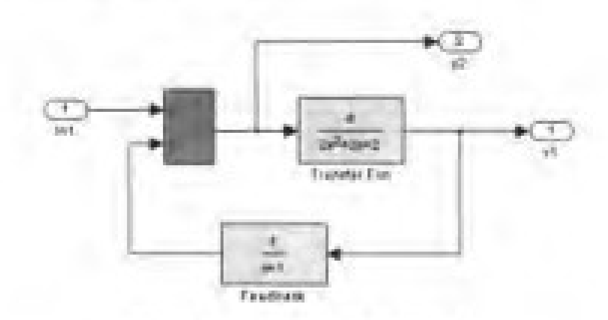


图9.135 非线性系统的模块

step 2 对非线性模块进行线性化操作，得到线性模块的参数。在MATLAB的命令窗口中输入下面的程序代码：

```
>> [A,B,C,D] = linmod('Sim40')
A =
```

```
   -1.5000   -1.0000    1.0000
    1.0000         0         0
         0    2.0000   -1.0000
B =
     1
     0
     0
C =
     0     2     0
     0     0     1
D =
     0
     1
```

step 3 将系统的状态空间转换为 LTI 对象。在 MATLAB 的命令窗口中输入下面的代码：

```
>> sys = ss(A,B,C,D)
a =
          x1    x2    x3
   x1   -1.5    -1     1
   x2      1     0     0
   x3      0     2    -1
 b =
        u1
   x1   1
   x2   0
   x3   0
 c =
        x1  x2  x3
   y1   0   2   0
   y2   0   0   1
d =
        u1
   y1   0
   y2   1
 Continuous-time model.
```



LTI(Linear Time-Invariant)对象是Simulink中主要的系统对象，可用Control System Toolbox中对应的函数来对转换后的LTI对象进行分析，关于该LTI对象的详细知识可以查看相应的帮助文件，这里就不详细介绍了。

step 4 绘制系统的波特相位振幅图。在 MATLAB 的命令窗口中输入下面的代码：

```
>> bode(sys)
```

step 5 查看图形结果。输入程序代码后，按 "Enter" 键，得到的图形如图 9.136 所示。



波特相位振幅图是LTI的振幅和相位变量的图表，其具体的物理含义比较复杂，关于bode命令的详细使用方法，请查看相应的帮助文件。

step 6 绘制系统的单位阶跃和脉冲响应图表。在命令窗口中输入下面的代码：

```
>> subplot(1,2,1);step(sys);grid
subplot(1,2,2);impulse(sys);grid
```

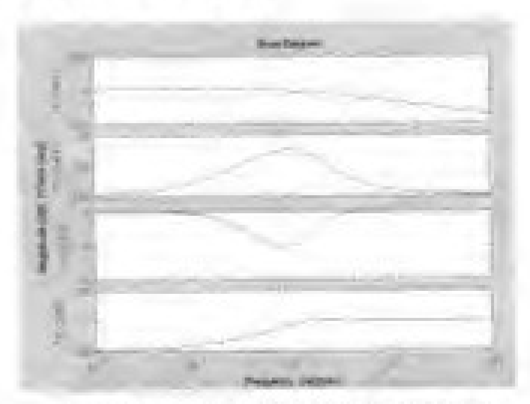


图 9.136　系统的波特相位振幅图

step 7 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图 9.137 所示。

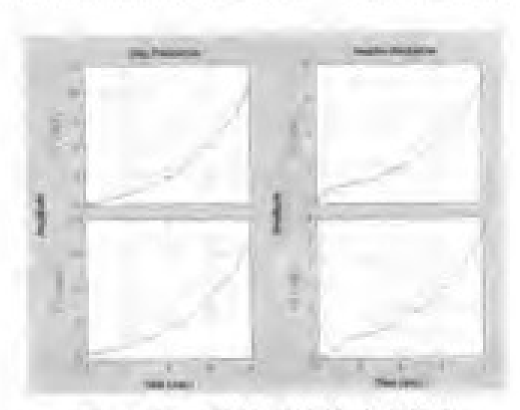


图 9.137　计算系统的响应信息

9.10.8 系统平衡点分析

在非线性系统的分析中，分析系统的稳定性或者稳态性状的时候需要分析系统的平衡点。在 Simulink中，所谓平衡点是指所有状态导数都等于零的点。如果只有部分状态导数是零，则成为偏平衡点。Simulink 通过使用trim命令来决定动态系统的稳定状态点，可以很方便地查看系统的平衡点性能，其具体的调用格式如下：

```
[x,u,y,dx] = trim('sys',x0,u0,y0,ix,iu,iy,dx0,idx,options,t)
```

在上面的调用格式中，只有第一个输入变量是必须的，其他的输入参数都是可选的。其中x0,u0,y0表示的是开始搜索(x,u,y)点的状态、输入和输出变量的初始数值，每一个分量的输入格式都必须是列向量的形式；ix,iu,iy则分别表示x0,u0,y0中保持不变的分量下标，options则是优化算法的参数选项数值。

例 9.16　演示如何使用 Simulink 来对系统的平衡点进行分析。

step 1 添加非线性系统的模块。为了分析系统的平衡点，首先需要添加系统模块，添加的模块如图 9.138 所示。

图 9.138 添加系统模块

step 2 设置命令搜索的初值条件。在 MATLAB 的命令窗口中输入下面的代码：

```
>> x = [ 0; 0; 0] ;
u = 0;
y = [ 1; 1] ;
ix = [] ;
iu = [] ;
iy = [ 1;2] ;
```

step 3 检测搜索点的平衡性。在命令窗口中输入下面的代码：

```
>> [ x,u,y,dx] = trim('Sim41',x,u,y,ix,iu,iy)
x =
     0
     0
     0
u =
     0
y =
     0
     0
dx =
     0
     0
     0
```

step 4 修改命令搜索的初值条件，并检测搜索点的平衡性能。在MATLAB的命令窗口中输入下面的代码

```
>> x = [ 0; 1; 0] ;
u = 0;
y = [ 3; 4] ;
ix = [] ;
iu = [] ;
iy = [ 1;2] ;
>> [ x,u,y,dx] = trim('Sim41',x,u,y,ix,iu,iy)
x =
   -0.0000
    0.2828
   -0.6000
u =
    1.0000
y =
   -0.6000
    0.4000
```

```
dx =
  1.0e-016 *
    0.5551
   -0.0000
         0
```



并不是所有求解平衡点的问题都会有解，如果无解，trim命令将会返回一个离期望状态偏差很少的一个解。

9.11 综合实例 1：交替执行系统

作为本章的最后，将使用Simulink来完成两个比较综合的实例。通过这些综合实例，读者可以加深对Simulink各种组件的认识，同时也会加强读者了解Simulink和实际问题相互联系的能力。为了加强对Simulink复杂模型创建过程的了解，本节依照步骤详细介绍各种实例的创建过程。

9.11.1 添加系统模块

在本小节中，将介绍如何在Simulink中创建交替执行系统，所涉及到的Simulink组件包括Merge模块、条件执行系统 、逻辑运算模块、Simulink中的Boolean信号、交替执行系统和利用程序修改模块的背景颜色等各种内容。

例 9.17 使用Simulink来创建交替执行系统，在不同时段中显示两种不同的信号图形，最后完成的综合信号图形如图9.139所示。



图 9.139 交替执行系统的图形

在上面的图形中，在0～5s的时间段内显示的是正弦函数图形，在5～10s的时间段内显示的则是周期折线图形；在另一个10s内，将重新显示上面的图形。下面将分步骤详细介绍上面图形的创建过程。

step 1 创建系统模块。在本小节中所创建的系统模块如图9.140所示。

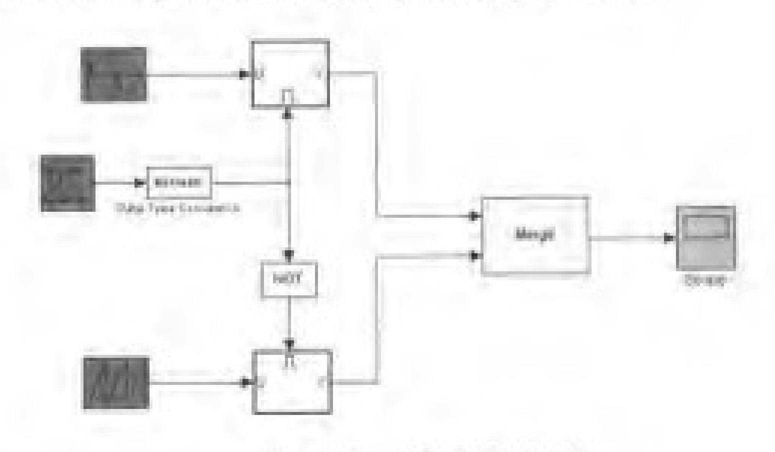


图 9.140 添加的系统模块

step 2 系统模块功能分析。由于系统模块比较复杂，为了让读者更好地了解系统模块的功能，在本步骤中将简要介绍模块的功能。

- **输入信号：**在系统的左侧有两个信号输入模块："Repeating Sequence"模块和正弦波形模块。两个模块可以产生标量输出信号，其形式和模块上的图标相同。
- **条件执行系统：**上面两个不同的输入信号分别和两个不同的"Enabled"子系统相连接，两个子系统的控制信号是由"Discrete Pulse Generator"模块产生的离散信号，该信号将以 5s 为周期，数值为 0 或者 1。当控制信号输出数值为 1 时，系统将会触发上面的子系统；当控制信号输出为 0 时，系统将会触发下面的子系统。
- **Merge 模块：**两个子系统输出的信号作为 Merge 模块的输入信号，产生一个综合后的信号，在 Scope 模块中显示。

9.11.2 设置系统模块的属性

延续上面小节的步骤。

step 1 设置"Sine Wave"模块的属性。在本系统中，输入信号的属性将会影响整个系统的输出结果，因此需要设置对应的属性，如图 9.141 所示。



图 9.141　设置正弦波的属性

在上面的对话框中，将正弦波的振幅设置为 1，频率设置为 pi，将 phase 和 bias 都设置为 0，得到周期为 2s 的正弦波图形。

step 2 设置"Repeating Sequence"模块的属性。作为另外一个输入信号，需要设置"Repeating Sequence"模块产生信号的属性，如图 9.142 所示。

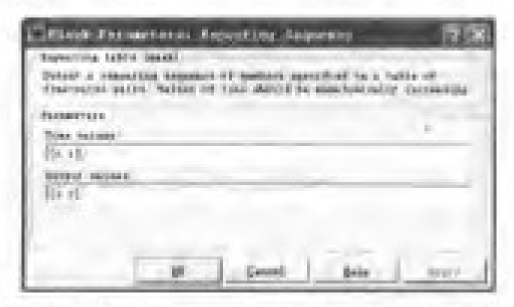


图 9.142　设置 Repeating Sequence 信号的属性

step 3 分析"Repeating Sequence"模块的属性。由于是第一次使用到"Repeating Sequence"模块，这里将详细介绍该信号的属性特征。在Simulink中，模块将会输出用户自行设定的波形，而且输出的信号为标量信号。可以在属性对话框中设置参数"Time values"和"Output values"的数值，设置输出信号的属性。其中参数"Times values"指定样本时间变量，参数"Output values"指定对应信号时间内的信号振幅。



> 实质上，参数"Times values"也等于指定了波形的周期时间。对于样本时间内的数据点，"Repeating Sequence"模块将会使用线性插值的方法计算数值。

在本实例中，指定的时间参数为[0 1]，输出数值参数为[0 2]，将会产生一个周期为1s的折线波形，振幅为2。

step 4 查看"Repeating Sequence"的子系统模块。从上面的对话框中可以看出该模块本身是经过封装的子系统模块，其对应的子系统如图9.143所示。

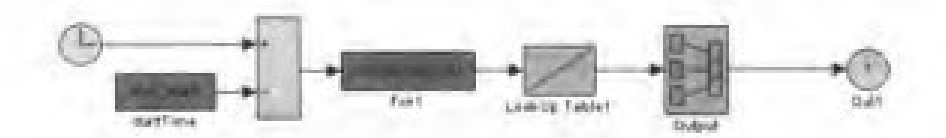


图9.143 模块对应的子系统模块

step 5 查看"Repeating Sequence"模块的源代码。在上面的系统中，startTime和Fcn1模块对应的都是S函数模块，其中startTime模块是由C语言编写并经过编译的文件，其具体代码如下：

```
/*
* File : sfun_tstart.c
#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME  sfun_tstart
#include "simstruc.h"
/* Function: mdlInitializeSizes ===========================================
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* Return if number of expected != number of actual parameters */
        return;
    }
    if (!ssSetNumInputPorts(S, 0)) return;
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 1);
    ssSetOutputPortDataType(S, 0, SS_DOUBLE);  /* same as clock block */
    ssSetOutputPortConstOutputExprInRTW(S, 0, 1);
    ssSetNumSampleTimes(S, 1);
    ssSetOptions(S,
                 SS_OPTION_WORKS_WITH_CODE_REUSE |
                 SS_OPTION_USE_TLC_WITH_ACCELERATOR);
}
/* Function: mdlInitializeSampleTimes =====================================
static void mdlInitializeSampleTimes(SimStruct *S)
```

```
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
    ssSetModelReferenceSampleTimeDefaultInheritance(S);
}
/* Function: mdlStart ===================================================
#define MDL_START
static void mdlStart(SimStruct *S)
{
    real_T *y = (real_T *)ssGetOutputPortSignal(S,0);
    *y = ssGetTStart(S);
}
/* Function: mdlOutputs =================================================
static void mdlOutputs(SimStruct *S, int_T tid)
{
    /* start time set in mdlStart, never changes */
}

/*================================*
 * Required S-function trailer *
 *================================*/
#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

由于上面的代码是经过C语言编译后完成的，超过了本书讨论的范围，这里就不详细分析了。之所以列出上面的代码，是为了方便读者理解模块产生的原理。同时，由于该模块是经过封装的模块，可以在封装属性对话框中设置模块的初始化属性，如图 9.144 所示。

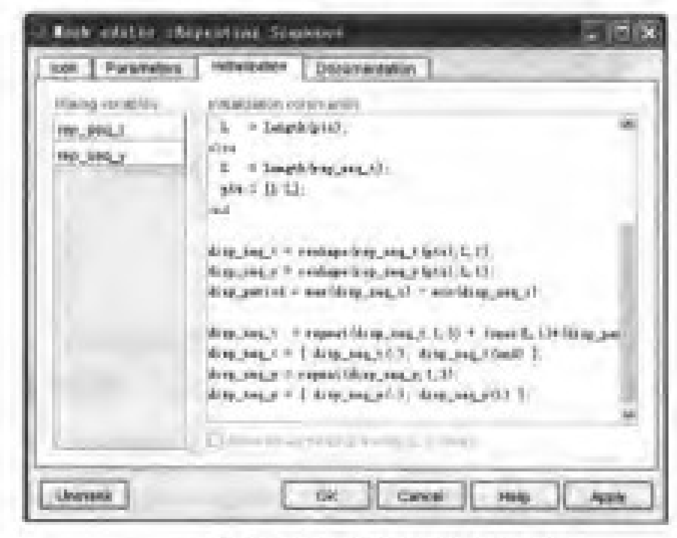


图 9.144　模块的初始化属性



关于该模块的其他属性，由于比较复杂，在这里就不详细展开分析了，感兴趣的读者可以查看关于该模块的帮助文件。

step 6 设置控制信号模块的属性。在本实例中，控制信号是离散信号，需要为其设置具体的参数属性，如图 9.145 所示。



图 9.145　设置控制信号模块的属性

在上面的对话框中，将 "Pulse type" 参数设置为 "sample-based"。这样，该模块将会产生一个用户指定的波形间隔输出离散波形。

step 7 设置 "Data Type Conversion" 模块的属性。由于控制信号的功能是触发子系统，因此需要将输出信号转换为 "Boolean" 数据类型，该模块的属性如图 9.146 所示。

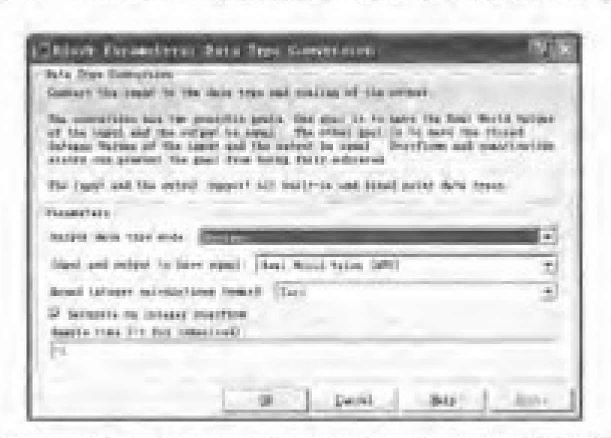


图 9.146　设置 "Data　Type　Conversion" 模块的属性

step 8 设置 Simulink 中的 Boolean signals 属性。选择模块编辑对话框中的 "Simulation" ⇨ "Configuration Parameters" 命令，打开 "Configuration Parameters" 对话框，选择其中的 "Optimization" 选项，在对话框的右侧选中 "Implement logic signals as boolean data" 选项，如图 9.147 所示。



由于在后面的步骤中需要将上面的Boolean信号作为Logical Operator模块的输入信号，需要在 Simulink 中对 Boolean signals 进行属性设置。



图 9.147 设置 Simulink 的属性

9.11.3 添加“Enabled”子系统

延续上面小节的步骤。

step 1 设置“Enabled”子系统的模块。双击上面的子系统模块，打开模块编辑器，在其中添加子系统的模块，如图 9.148 所示。

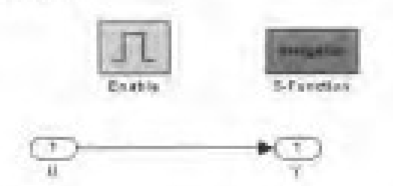


图 9.148 添加子系统模块

step 2 设置“Enable”模块的属性。双击上面系统中的“Enable”模块，打开对应的属性对话框，设置的属性如图 9.149 所示。



图 9.149 设置“Enable”模块的属性

step 3 设置“S-Function”模块的属性。双击上面系统中的“S-Function”模块，打开对应的属性对话框，设置属性如图 9.150 所示。

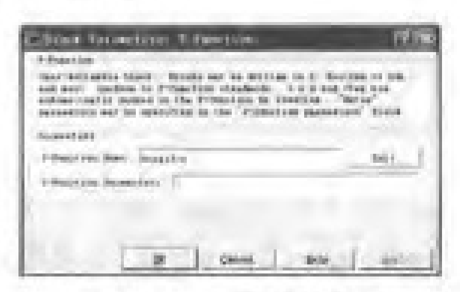


图 9.150 设置模块属性

step 4 添加S函数的程序代码。单击上面对话框中的“Edit”按钮，查看S函数“mergefcn”的具体代码如下：

```
function [ sys,x0,str,ts] = mergefcn(t,x,u,flag)
% S-Function for Simulink merge demonstration.
switch flag,
    case 0
    [ sys,x0,str,ts] =mdlInitializeSizes;
  case 2
    sys=mdlUpdate(t,x,u);
  case 9
    sys = mdlTerminate;
  case { 1, 3, 4 }
    sys=[ ];
  otherwise
    error([ 'Unhandled flag = ',num2str(flag)]);
end
%==========================================================================%
function [ sys,x0,str,ts] = mdlInitializeSizes()
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 0;
sizes.NumInputs      = 0;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
str = [];
x0  = [];
ts  = [-1 0];   % inherited sample time
%==========================================================================%
function sys = mdlUpdate(t,x,u)
root = get_param(bdroot,'Handle');
subs = find_system(root,'Tag','MergeExample');
parent = get_param(get_param(gcbh,'Parent'),'Handle');
notme = subs(find(subs ~= parent));
me = subs(find(subs == parent));
if ~strcmp(get_param(me,'BackgroundColor'),'green')
  set_param(me,'BackgroundColor','green');
  drawnow
  set_param(notme,'backgroundcolor','white')
end
shortpause
sys = [];
function sys = mdlTerminate
root = get_param(bdroot,'Handle');
subs = find_system(root,'Tag','MergeExample');
set_param(subs,'Backgroundcolor','white')
sys = [];
function shortpause
pause(0.1)
```

上面程序代码主要功能是实现两个信号的融合，读者可以参考前面章节中关于S函数的内容来分析上面的代码，这里主要分析上面的mdlUpdate函数，其程序代码如下：

```
function sys = mdlUpdate(t,x,u)
root = get_param(bdroot,'Handle');
subs = find_system(root,'Tag','MergeExample');
parent = get_param(get_param(gcbh,'Parent'),'Handle');
notme = subs(find(subs ~= parent));
me = subs(find(subs == parent));
if ~strcmp(get_param(me,'BackgroundColor'),'green')
  set_param(me,'BackgroundColor','green');
  drawnow
  set_param(notme,'backgroundcolor','white')
end
shortpause
sys = [];
```

上面的程序代码中，其主要的功能是动态设置两个子系统背景颜色。在程序代码中，根据子系统是否运行，使用下面的程序代码设置BackgroundColor属性：

```
set_param(me,'BackgroundColor','green');
```

其他程序代码和上面代码的功能类似，读者可以自行分析。

step 5 设置"NOT"模块的属性。双击子系统的"NOT"模块，打开属性对话框，在其中设置模块的属性，如图9.151所示。

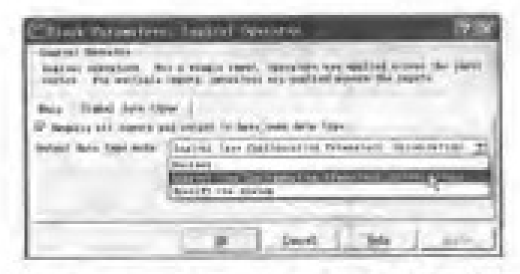


图9.151　设置"NOT"模块的属性



在上面的对话框中，选择"Signal data types"选项卡，在"Output data type mode"选框中选择"Boolean"选项，设置该模块的输出信号的属性为Boolean。

step 6 设置"Merge"模块的属性。双击子系统的"Merge"模块，打开属性对话框，在其中设置模块的属性，如图9.152所示。

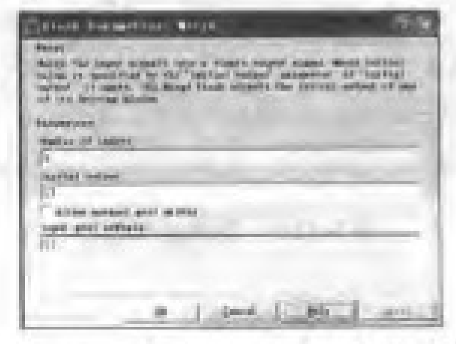


图9.152　设置"Merge"模块的属性

9.11.4 运行仿真系统

延续上面小节的步骤。

step 1 运行系统。将系统的仿真时间设置为 30，运行仿真系统。当运行上面的子系统模块时，系统的模块如图 9.153 所示。

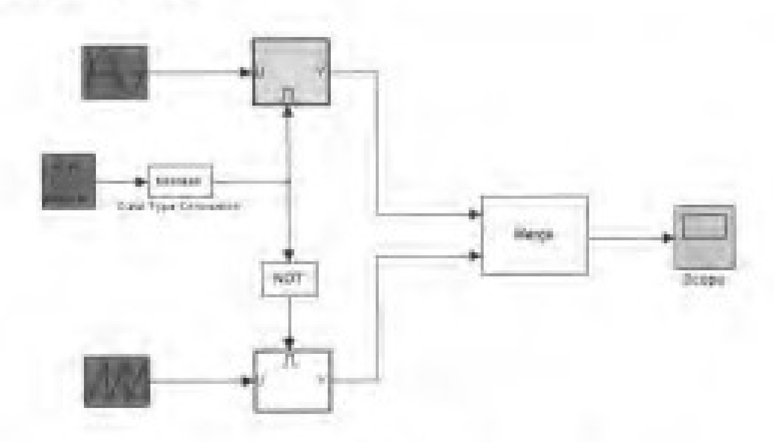


图 9.153 执行上面的子系统

当程序运行下面的子系统模块时，系统的模块如图 9.154 所示。

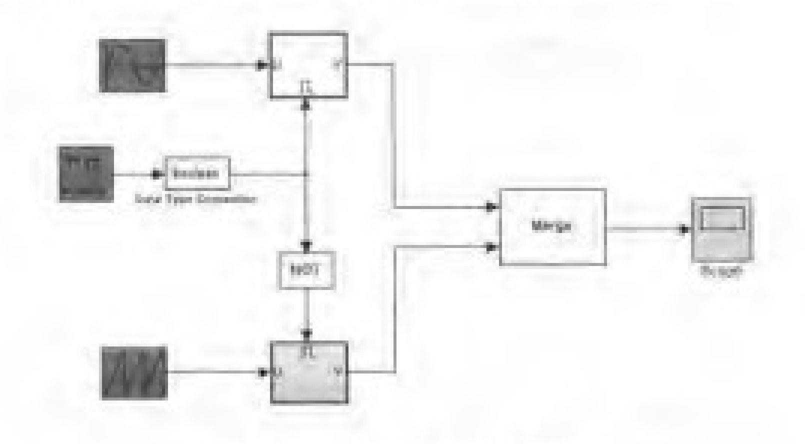


图 9.154 执行下面的子系统

step 2 查看仿真结果。同时，得到的仿真图形如图 9.155 所示。



图 9.155 仿真结果图形

9.12 综合实例 2：雷达轨迹分析

本节将介绍一个综合的实例，分析雷达系统绘制的飞机运动的轨迹。和上面类似，本综合实例

需要运用Simulink中的封装子系统、S函数运算等各种常见功能，由于模型创建比较复杂，下面将分小节详细介绍。

9.12.1 系统模块简介

例 9.18 使用Simulink来创建雷达飞行轨迹系统，并根据仿真得到的结果绘制各种参数的图形，其中雷达获取的飞机运动轨迹和模拟的运动轨迹如图9.156所示。

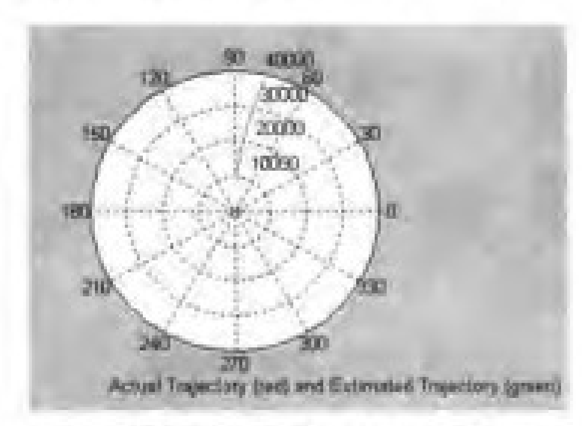


图 9.156 雷达获取的飞机轨迹

同时，为了分析系统模拟的精度，需要分析实际轨迹和模拟轨迹的偏差程度，需要绘制偏差程度的图形，如图9.157所示。

最后，得到飞机在不同方向上运行轨迹的对比情况如图9.158所示。

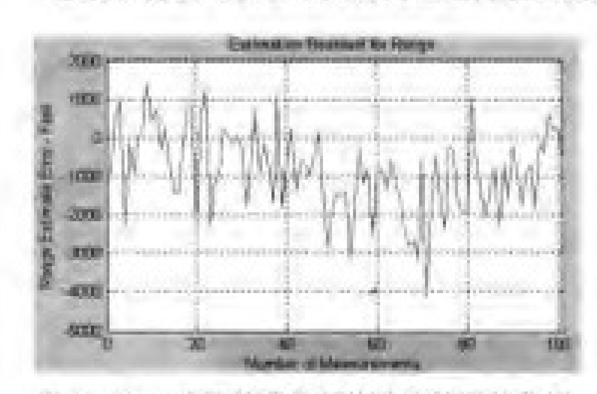


图 9.157 实际轨迹和模拟轨迹的误差曲线

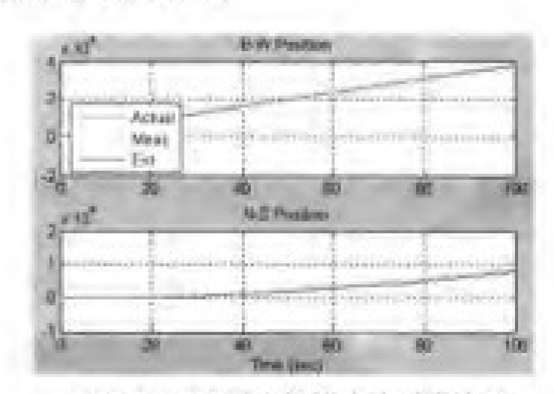


图 9.158 不同方向轨迹的对比情况

9.12.2 添加系统模块

下面将详细分析上面系统的创建过程。

step 1 创建系统模块。在本小节中所创建的系统模块如图9.159所示。

step 2 分析系统模块功能。由于这个系统比较复杂，在详细介绍系统各个模块的属性和功能之前，首先介绍该系统模块的整体功能。

- 求解飞机在笛卡儿系运动轨迹坐标。在"Cartesian to Polar"模块左侧的系统模块，其主要功能是求解飞机在笛卡儿坐标系下的轨迹坐标数值。首先通过"Random aircraft motion"模块产生的随机数值作为飞机运动的初始信号，将该信号分别通过"Cross-Axis

Acceleration Model"和"Thrust-Axis Accleration Mode"子系统模块，输出飞机横轴(Cross-Axis)和推力轴(Thrust-Axis)运动的加速度，然后将上面的加速度数值信号分别通过两次积分器模块，得出飞机在横轴(Cross-Axis)和推力轴(Thrust-Axis)方向上的运动位移数值。最后将两个位移数值通过"Actual Position"模块，传递到MATLAB的工作空间中。

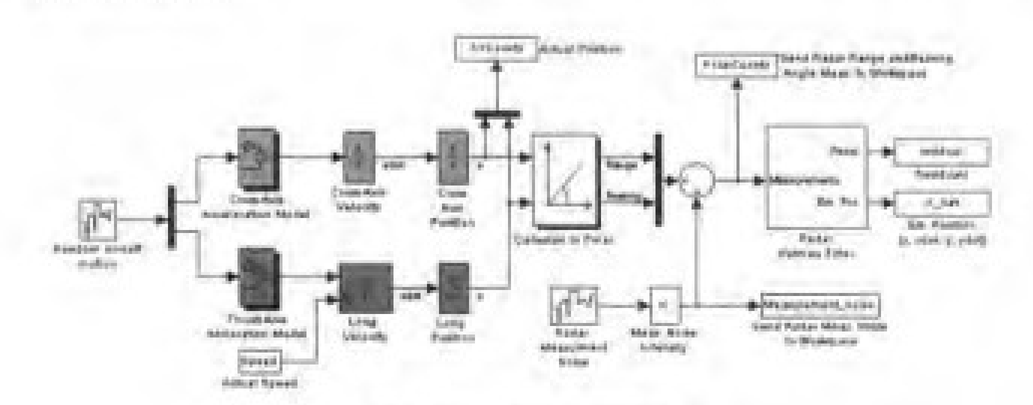


图9.159　创建系统模块

- 将笛卡儿系的坐标转换为极坐标。将上面系统计算出来的飞机运动轨迹的XY坐标，通过"Cartesian to Polar"模块转换为极坐标下的数值Range(极径)和Bearing(极角)。
- 计算飞机运动轨迹的极坐标。将上面模块转换得到的极坐标数值加上使用雷达系统的测量误差数据，得出实际使用雷达测量得到飞机运动轨迹的极坐标，并将得到的转换数据通过"Send Radar Range andBearing Angle Meas. to Workspace"模块传递到工作空间中。
- 将极坐标数据进行"Kalman Filter"处理，得到处理后的数据。将上面步骤得到的极坐标数据，经过"Radar Kalman Filter"模块得出经过"Kalman Filter"处理后的数据，并分别通过"Residuals"和"Est. Position"模块传递到工作空间中。

step 3 设置系统模块的属性。由于系统模块繁多，为了方便查看系统采样的层次，可以选择模块编辑对话框中的"Format"⇨"Prot/Signal Displays"⇨"Sample Time colors"命令，当运行仿真系统后，不同采样时间的模块会显示不同颜色，如图9.160所示。

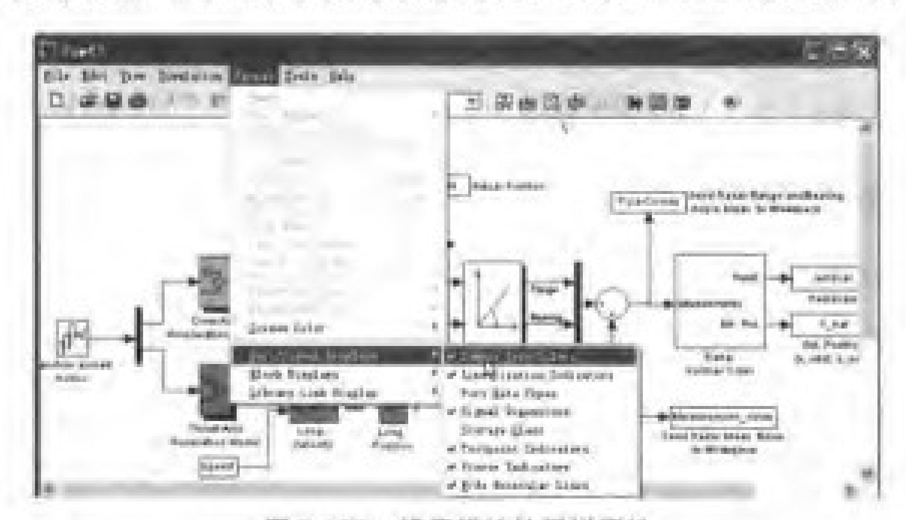


图9.160　设置模块的采样属性

step 4 设置系统的解法器。选择模块编辑对话框中的"Simulation"⇨"Configuration Parameters"命令，打开"Configuration Parameters"对话框，选择其中的"Solver"选项，在对话框

的右侧 "solver type" 选项卡中的 "type" 选框中选择 "Fix-step"，在 "solver" 选框中选择 "ode5(Dormand-Prince)" 选项，如图 9.161 所示。

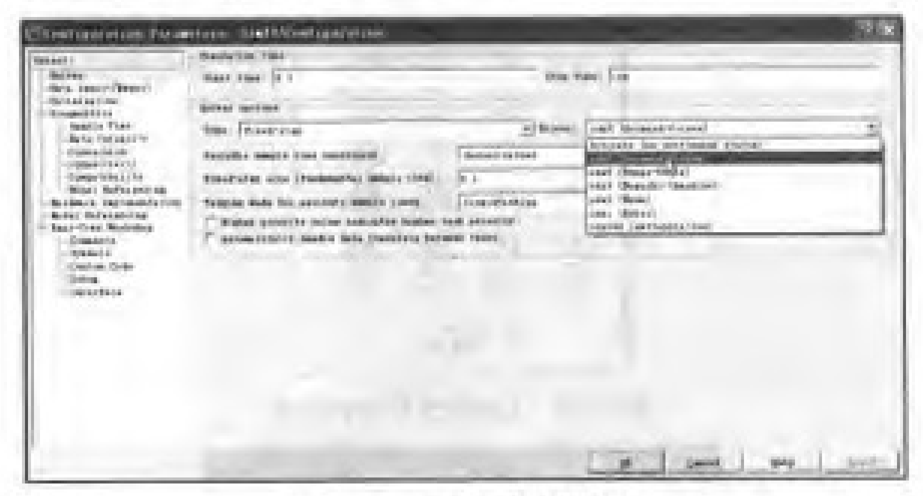


图 9.161　设置系统的解法器



关于系统解法器的具体选择问题，由于涉及到具体的微分或者差分方程的内容，在这里就不详细介绍了，感兴趣的读者可以参看相应的帮助文件。

step 5 设置"Random aircraft motion"模块的属性。双击上面系统中的"Random aircraft motion"模块，打开对应的属性对话框，在其中设置其属性，如图 9.162 所示。

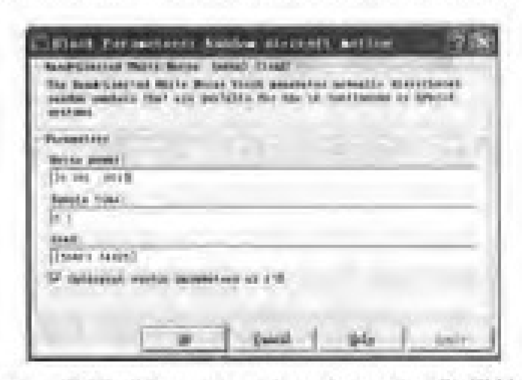


图 9.162　设置 "Random aircraft motion" 模块的属性

在本实例中 "Random aircraft motion" 模块的功能是产生飞机运动的随机坐标数值。根据上面的对话框，该模块也是被封装后的模块。选择该模块，单击鼠标右键，在弹出的快捷菜单中选择 "Link Options" ⇨ "Go To Library Block" 命令，打开 Simulink 中的模块库，可以查看该模块在模块库中位置，如图 9.163 所示。

如果编辑该模块，则首先选择该模块，单击鼠标右键，在弹出的快捷菜单中选择 "Link Options" ⇨ "Disable Link" 命令，取消该模块和模块库的关联。然后在需要选择的模块，单击鼠标右键，在弹出的快捷菜单中选择 "Edit Mask" 选项，打开关于该封装模块的编辑器。

例如，如果希望修改该模块的初始值，可以选择编辑器中的 "Initialization" 选项卡，在其中设置模块的初始数值，如图 9.164 所示。

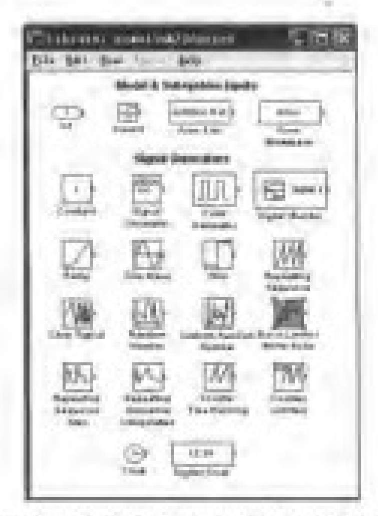


图 9.163　Simulink 中的模块库



图 9.164　设置模块的初始数值

9.12.3 添加"Cross-Axis Acceleration Model"子系统

延续上面小节的步骤。

step 1 添加"Cross-Axis Acceleration Model"子系统的模块。双击系统中的"Cross-Axis Acceleration Model"子系统，打开模块编辑框，添加对应的子系统，如图 9.165 所示。

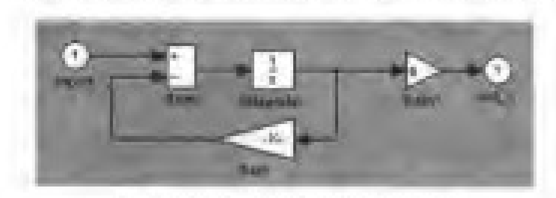


图 9.165　添加子系统模块

step 2 分析上面系统模块功能。根据 Simulink 的基础知识，上面的系统模块相当于下面的微分方程组：

$$\begin{cases} I - Kx = x' \\ \alpha = gx \end{cases}$$

其中，I 表示前面步骤中输入的随机信号，K 表示微分方程的参数，x 表示微分方程的中间变量，α 则是模块输出的横轴加速度。



上面系统求解的是飞机运动在横轴方向上的加速度，"Thrust-Axis Accleration Mode" 子系统模块和该模块在结构上完全相同，只是参数 K 在数值上不同。

9.12.4 添加 "Cartesian to Polar" 子系统

延续上面小节的步骤。

step 1 添加 "Cartesian to Polar" 子系统的模块。双击上面系统中的 "Cartesian to Polar" 子系统，打开模块编辑框，添加对应的子系统，如图 9.166 所示。

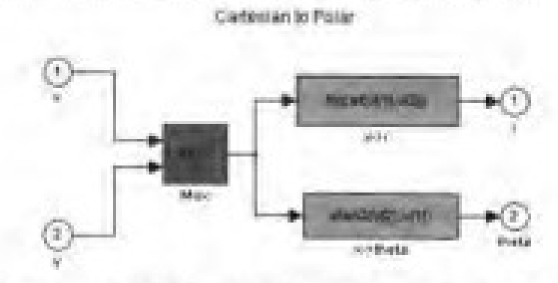


图 9.166 添加 "Cartesian to Polar" 子系统的模块

在上面的系统模块中，hypot 和 atan2 都是 MATLAB 中内置的数学操作函数，分别将输入的 x 和 y 坐标数值转换为极坐标下的 r 和 θ，其具体的计算公式如下：

$$\begin{cases} r=\sqrt{x^2+y^2} \\ \theta=\alpha\tan(\dfrac{y}{x}) \end{cases}$$

step 2 封装 "Cartesian to Polar" 子系统的模块。选择 "Cartesian to Polar" 子系统模块，然后单击鼠标右键，在弹出的快捷菜单中选择 "Mask Subsystem" 选项，打开模块编辑器，选择 "Icon" 选项卡，在其中设置封装子系统的图标，如图 9.167 所示。



图 9.167 设置封装子系统的图标

step 3 设置 "Radar Measurment Noise" 模块的属性。双击系统中的 "Radar Measurment Noise" 模块，打开属性对话框，在其中设置对应的属性，如图 9.168 所示。

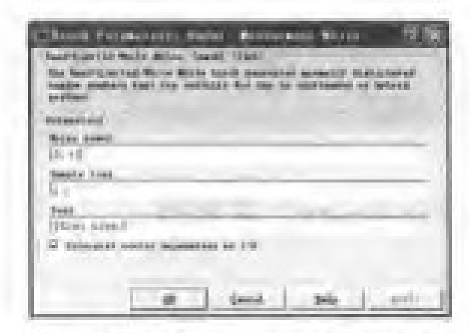


图 9.168　设置"Radar Measurment Noise"模块的属性



关于该模块的属性设置，和前面步骤中设置"Random aircraft motion"模块的属性类似，这里就不重复介绍了。

step 4 设置"Meas. Noise Intensity"模块的属性。双击系统中的"Meas. Noise Intensity"模块，打开属性对话框。在其中设置对应的属性，如图 9.169 所示。



图 9.169　设置模块的参数值

在上面的对话框中，设置该模块的参数值 K 和前面步骤中加速度的参数 K 相同。为了后面步骤的操作方便，有必要对该模块进行封装。

step 5 封装"Meas. Noise Intensity"模块。选择"Meas. Noise Intensity"子系统模块，然后单击鼠标右键，在弹出的快捷菜单中选择"Mask Subsystem"选项，打开模块编辑器，选择"Parameters"选项卡，设置封装子系统的参数，如图 9.170 所示。



图 9.170　设置封装模块的参数

对于其他封装模块的参数值，限于篇幅，这里就不一一列出了，主要是设置对应的图标、文字提示等属性，完成后的封装系统如图9.171所示。



图9.171 完成后的封装子系统

9.12.5 添加"Radar Kalman Filter"子系统

延续上面小节的步骤。

step 1 添加"Radar Kalman Filter"子系统的模块。双击系统中的"Radar Kalman Filter"模块，打开模块编辑器，在其中添加子系统的模块，如图9.172所示。

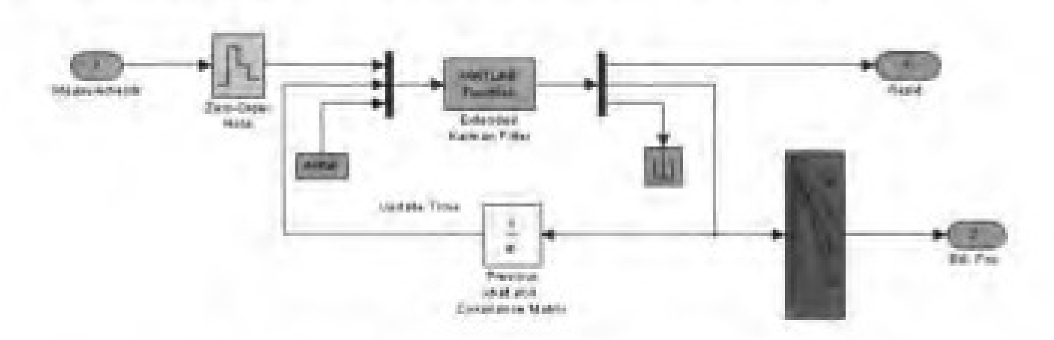


图9.172 添加"Radar Kalman Filter"子系统的模块

step 2 设置"Zero-Order Hold"模块的属性。该系统的主要功能是将极坐标数据经过Kalman Filter处理，双击"Zero-Order Hold"模块，打开对应的属性对话框，在其中设置其具体属性，如图9.173所示。



图9.173 设置"Zero-Order Hold"模块的属性



在Simulink中，"Zero-Order Hold"（简称为ZOH）模块的主要功能是实现离散信号和连续信号之间的转换，关于该模块的详细信息请自行查看帮助文件。

step 3 设置"Extended Kalman Filter"模块的属性。双击系统中的"Extended Kalman Filter"模块，打开属性对话框，设置该模块的属性，如图9.174所示。



图9.174 设置模块的属性

step 4 查看程序代码。在对话框中的"MATLAB Function"选框中输入"aero_extkalman"，这是M文件的名称，其具体的代码如下：

```
function funcout = aero_extkalman pinput)
%  See the description in the "Extended Kalman Filter" Brochure for
%  the equations.
% Initialization
meas = pinput(1:2);
xhatPrev = pinput(3:6);
PPrev = pinput(7:22);        %  Covariance matrix (zeros assumes perfect
estimate)
deltat = pinput(23);
xhat = xhatPrev(:);            %  Estimate
P    = reshape(PPrev,4,4);
%  Radar update time deltat is inherited from workspace where it was
defined by raddat.
% 1. Compute Phi, Q, and R
Phi = [1 deltat 0 0; 0 1 0 0 ; 0 0 1 deltat; 0 0 0 1];
Q =  diag([0 .005 0 .005]);
R =  diag([300^2 0.001^2]);
% 2. Propagate the covariance matrix:
P = Phi*P*Phi' + Q;
% 3. Propagate the track estimate::
xhat = Phi*xhat;
% 4 a). Compute observation estimates:
Rangehat = sqrt(xhat(1)^2+xhat(3)^2);
Bearinghat = atan2(xhat(3),xhat(1));
% 4 b). Compute observation vector y and linearized measurement matrix
M
yhat =   [Rangehat
          Bearinghat];
M = [ cos(Bearinghat)            0 sin(Bearinghat)            0
   -sin(Bearinghat)/Rangehat 0 cos(Bearinghat)/Rangehat 0 ];
% 4 c).  Compute residual (Estimation Error)
residual = meas - yhat;
% 5. Compute Kalman Gain:
W = P*M'/(M*P*M'+ R);
% 6. Update estimate
xhat = xhat + W*residual;
% 7. Update Covariance Matrix
P = (eye(4)-W*M)*P*(eye(4)-W*M)' + W*R*W';
% Output columwise for Simulink.
```

```
funcout = [ residual;xhat;P(:);deltat];
```

在上面的程序代码中，依次按照步骤计算 Extended Kalman Filter 中的转换工作，得到的结果就是将极坐标数值转换为 Kalman Filter 处理后的数据。

添加程序代码

延续上面小节的步骤。

step 1 编写系统参数的 M 文件，其具体代码如下：

```
%RADDAT 是运行仿真系统所需要的基础参数值
g = 32.2;      % 加速度（相对于重力加速度）
tauc = 5;      % 横轴加速的时间
tauT = 4;      % 推力轴加速的时间
Speed = 400; % 在 y 方向上的初始速度
deltat = 1;
```

上面的数据都是仿真系统的基础参数数据，将上面的代码保存为“aero_raddat.m”文件，在后面的步骤中将在运行仿真系统之前加载这些数据。

step 2 编写绘制参数图形的 M 文件，其具体代码如下：

```
%RADPLOT
%delt = 0.1;   %   仿真的样本时间
%deltat = 5;   %   雷达更新的时间
% Post Processing of the Data for Plotting:
% 在极坐标条件下绘制飞机轨迹
pos = [ 10 40 500 300];
h_1 = figure(1);
set(h_1,'pos',pos);
polar(PolarCoords(:,2) - Measurement_noise(:,2), ...
      PolarCoords(:,1) - Measurement_noise(:,1),'r')
hold on
rangehat = sqrt(X_hat(:,1).^2+X_hat(:,3).^2);
bearinghat = atan2(X_hat(:,3),X_hat(:,1));
polar(bearinghat,rangehat,'g')
text(-35000,-50000,'Actual Trajectory (red) and Estimated Trajectory
(green)')
% 创建新的图形窗口，绘制误差曲线
h_2 = figure(2);
set(h_2,'pos',[ pos(1)+500 pos(2) pos(3:4)]);
plot(residual(:,1)); grid;set(gca,'xlim',[ 0 length(residual)]);
xlabel('Number of Measurements');
ylabel('Range Estimate Error - Feet')
title('Estimation Residual for Range')
% 创建新的图形窗口，绘制对比情况
h_3 = figure(3);
set(h_3,'pos',[ pos(1) pos(2)+350 pos(3:4)]);
XYMeas = [ PolarCoords(:,1).*cos(PolarCoords(:,2)), ...
          PolarCoords(:,1).*sin(PolarCoords(:,2))];
t_full = [ 0:0.1:100]';
t_hat = [ 0:deltat:100]';
subplot(211)
plot(t_full,XYCoords(:,2),'r');
```

```
grid on;hold on
plot(t_full,XYMeas(:,2),'g');
plot(t_hat,X_hat(:,3),'b');
title('E-W Position');
legend('Actual','Meas','Est',3);
hold off
subplot(212)
plot(t_full,XYCoords(:,1),'r');
grid on;hold on
plot(t_full,XYMeas(:,1),'g');
plot(t_hat,X_hat(:,1),'b');
xlabel('Time (sec)');
title('N-S Position');
hold off
```



在上面的程序代码中，依次利用仿真系统传递到MATLAB工作空间中的变量，绘制各种参数的图形，关于该代码的主要分析工作交给读者自行完成。

step 3 加载上面的程序代码。在模块编辑器中单击鼠标右键，在弹出的快捷菜单中选择"Model Properties"选项，打开"Model Properties"对话框，选择"Callbacks"选项卡，在该选项卡对应的选框中加载上面的程序代码，如图 9.175 所示。



图 9.175　设置系统模块的属性

9.12.7 运行仿真系统

延续上面小节的步骤。

step 1 运行系统仿真，得到仿真结果。将系统仿真时间设置为 100，然后运行系统，得到的仿真结果如图 9.176 所示。

step 2 查看运行后的系统模块。由于用户设置了不同采样模块为不同的颜色，因此运行后的系统模块如图 9.177 所示。

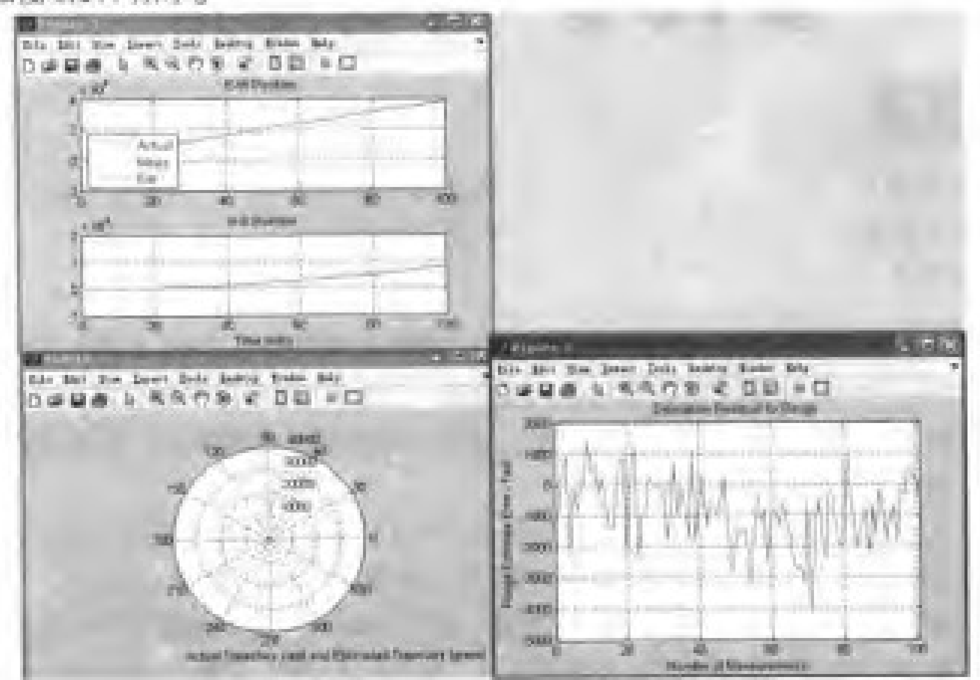


图 9.176　得到的仿真结果

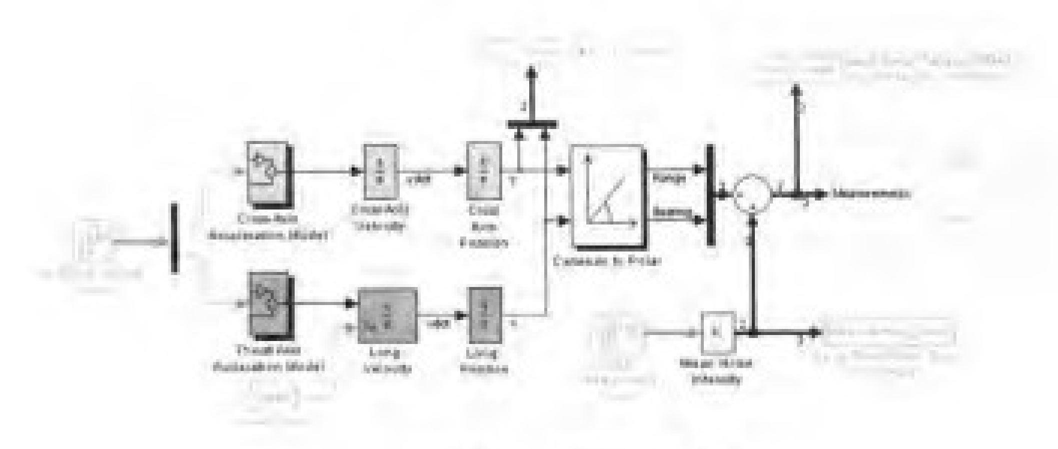


图 9.177　运行的系统模块

9.13 小结

在本章中，主要向读者介绍了 Simulink 中的封装子系统、使能子系统、触发子系统和 S 函数等高级技术，结合前面的内容，使得读者可以对 Simulink 的知识有一个全面的了解。这些高级技术使得 Simulink 可以更加方便地和实际工程结合，实现各种复杂功能，希望读者能够熟练掌握。后面，将向读者介绍句柄图形的内容。

◆　◆　◆

第10章　句柄图形

本章包括

- 句柄图形体系
- 图形句柄的操作
- 图形对象的操作
- 高层绘图命令
- 坐标轴对象

在前面的章节中，已经了解了使用MATLAB绘制二维和三维图形的方法。但是，在介绍绘制方法的时候，使用的都是MATLAB的高层绘图命令，当希望"个性化"图形属性的时候，高层绘图命令就会有很大的局限。这个时候，句柄图为用户提供强有力的功能。在本章中，将详细介绍如何使用MATLAB中的句柄图形绘制各种复杂图形。

本章的内容将更深入MATLAB可视化功能的内核，可以更深入地理解高层绘图命令，进而更精确更生动地理解图形，还可以利用底层绘图命令和图形对象属性开发专用的绘图函数。

在本章中，将详细介绍句柄图形体系、图形对象、属性和操作方法，尽量做到由表及里、由浅入深，同时尽量做到突出要点和难点，对各个对象最常用的、最不可或缺的、较难把握的属性进行详细介绍。同时，在本章的最后，将向读者介绍关于句柄图形的综合案例，从中了解到句柄图形的使用方法。

10.1 句柄图形体系

句柄图形是对底层图形例程集合的总称，它实际上进行生成图形的工作。这些细节通常隐藏在图形M文件的内部，但如果想使用它们也是可得到的。句柄图形可以被任何人用来改变MATLAB生成图形的方式，不论只想在一幅图里做一点小变动，还是想做影响所有图形输出的全局变动。句柄图形允许用户定制图形的许多特性，而这用高级命令和前几章里描述的函数是无法实现的。

底层绘图命令使用起来不像高层命令那样简单易懂，但是底层绘图命令直接操作基本绘图要素，可以更具个性化地表现图形。前面已经介绍过，本章不会介绍句柄图形的全部内容。这里的目的只是对句柄图形概念作基本了解，并提供足够多的信息，使得即使偶尔使用一下MATLAB的用户也可以利用句柄图形。



前面那些章提供的图形功能被认为是高级的命令和函数，包括plot、mesh、axis及其他。这些函数是建立在底层函数和属性的基础上，总称为句柄图形。

10.1.1 图形对象

前面多次提到句柄图形，在这里将详细介绍句柄图形的概念。在MATLAB中，每个图形的每一组成部分是一个对象，每一个对象有一系列句柄和它相关，而且每一个对象都有按需要可以改变的属性。

在计算机领域中，对象可以被粗略地定义为由一组紧密相关、形成唯一整体的数据结构或函数集合。在MATLAB中，图形对象是一幅图中很独特的成分，它可以被单独地操作。

具体来讲，由MATLAB的图形命令产生的每一件东西都属于图形对象，包括图形窗口或仅仅说是

图形，还有坐标轴、线条、曲面、文本和其他对象。



在MATLAB中生成的每个图形，都由多个图形对象组成，每个具体的图形不必包含全部对象，但是每个图形都必须具备根屏幕和图形窗。

10.1.2 句柄对象

在MATLAB中，每一个对象都有一个数字来标识，叫做句柄。当每次创建一个对象时，MATLAB就为它建立一个唯一的句柄。计算机屏幕作为根对象常常是0。Hf_fig=figure命令建立一个新的图形窗口，变量Hf_fig中返回它的句柄值。图形窗口的句柄为整数，通常显示在图形窗口标题条中。其他对象句柄是MATLAB满精度的浮点值。

所有产生对象的MATLAB函数（例如plot、mesh、surf及其他）都为所建立的每个对象返回一个句柄（或句柄的列向量）。有一些图形由一个以上对象组成。比如，一个网格图由一个曲面组成，它只有一个句柄；而waterfall图形由许多线条对象组成，每个线条对象都有各自的句柄。例如，命令行"Hl_wfall=waterfall(peaks(20))"对线条返回一个包含着20个句柄的列向量。



虽然句柄变量也是MATLAB变量中的一种，可以取任意名字，但是遵循某些命名规则使得能在M文件中很容易找到句柄变量。例如，可以将句柄变量以大写的H开头，跟以一个标识对象类型的字母，然后是一个或几个描述符。当对象类型不知道时，用字母x，比如Hx_obj。

10.1.3 句柄图形的结构

在MATLAB中，所有的图形对象按父对象和子对象组成层次结构。计算机屏幕是根对象，并且是所有其他对象的父亲。图形窗口是根对象的子对象；坐标轴和用户界面对象（在下一章讨论）是图形窗口的子对象；线条、文本、曲面、补片和图像对象是坐标轴对象的子对象，这种层次关系在图10.1中给出。

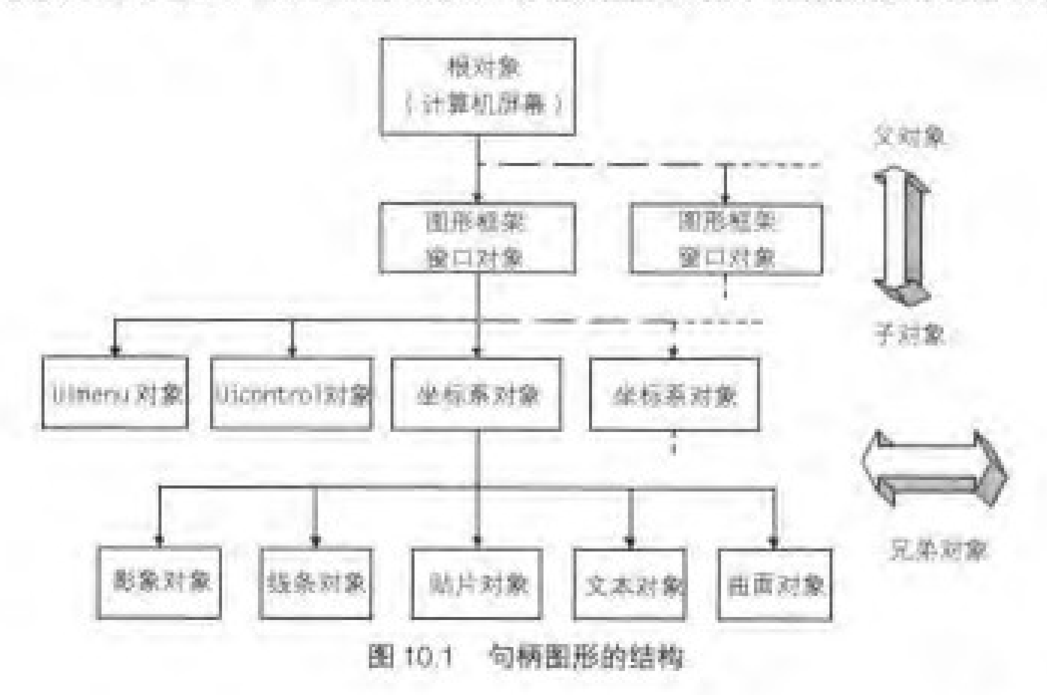


图10.1 句柄图形的结构

根可包含一个或多个图形窗口，每一个图形窗口可包含一组或多组坐标轴。所有其他的对象（除了uicontrol和uimenu外）都是坐标轴的子对象，并且在这些坐标轴上显示。所有创建对象的函数当父对象或对象不存在时，都会创建它们。例如，没有图形窗口，plot函数会用默认属性创建一个新的图形窗口和一组坐标轴，然后在这组坐标轴内画线。

10.1.4 图形对象的属性

所有对象都有一组定义其外貌和形状的属性，同时所有对象都有属性来定义它们的特征。尽管许多属性所有的对象都有，但与每一种对象类型（比如坐标轴、线、曲面）相关的属性列表都是独一无二的。对象属性包括对象的位置、颜色、类型、父对象、子对象及其他内容。每个不同对象都有和它相关的属性，可以改变这些属性而不影响同类型的其他对象。

对象属性包括属性名和与它们相关联的值。属性名是字符串，它们通常按混合格式显示，每个词的开头字母大写，比如"LineStyle"。但是，MATLAB识别一个属性时是不分大小写的。另外，只要用足够多的字符来唯一地辨识一个属性名即可。例如，坐标轴对象中的位置属性可以用"Position"，甚至"pos"来调用。

用户不仅可以查询当前对象的任意属性值，而且可以指定大多数属性的取值（在MATLAB7.0中，对象的有些属性只是只读的）。用户设定的属性值仅仅对特定的对象实例起作用，不会影响到不同对象、不同实例的属性。如果希望修改以后创建对象的属性值，可以设置属性的默认值，这样的修改就可以影响后面创建的所有对象。

当在MATLAB中创建图形对象时，并不需要指定对象的每个属性值，这是因为如果用户不指定属性值，MATLAB会对对象采用默认属性值。如果希望自己创建的图形具有统一的风格，可以设置一组"用户定义"的默认属性值。



在编写M文件的时候，可以使用属性名的缩写。但是笔者不建议使用这种缩写，以免带来不必要的麻烦。

10.2 图形句柄的操作

从前面内容中读者已经了解到图形对象的基础知识。本节将介绍如何对图形对象进行操作，包括创建图形句柄、获取图形对象的句柄、使用句柄操作图形对象等。

10.2.1 创建图形对象

在上面介绍的所有图形对象中，除了根屏幕之外，其他所有的图形对象都有相应的创建命令，这些创建命令名称都和相应的对象名称相同。例如，函数text会创建一个text对象，figure函数会创建一个figure对象，而且这些函数都是MATLAB的内置函数。每个命令在创建图形对象的同时，也会返回图形对象的句柄，用户可以使用该对象句柄来查询或者修改对象的属性值。

这些底层函数只能创建11种图形对象中的一个，并且将图形对象置于适当的父辈对象之中。例如，patch命令会利用当前坐标轴中的默认属性来创建方块对象，如果在执行该命令之前没有创建坐标轴、图形窗对象，则MATLAB会自动创建这些对象。如果在执行该命令之前已经有了这些对象，则方块对象将会绘制在之前的坐标轴对象中，而且不会影响坐标轴对象中的其他对象。



根据前面的介绍，如果使用高层绘图函数来绘制图形时，添加新的图形对象会将前面的操作全部覆盖，而使用底层绘图函数则不会造成这样的问题，只会在原来的基础上继续添加对象或者修改属性。

表 10.1 列出了 MATLAB7.0 中所有图形对象的创建函数。

表 10.1 MATLAB 中的图形对象创建函数

函数	功能
Axes	创建图形的坐标轴对象
Figure	创建或者显示图形窗口对象
Image	使用颜色映射表索引或 RGB 数值的二位图像
Light	位于坐标轴中，能够影响曲面或者曲片的有方向的光源
Line	由顺序链接坐标数据的直线线段构成的线条
Patch	将矩阵的每列数据构成多边形的小面，创建一个块对象
Rectangle	矩形或者椭圆形的二维填充图案，创建一个方对象
Surface	由矩阵数据定义的矩阵创建而成的曲面对象
Text	创建位于坐标轴系统内的字符串对象
Uicontrol	创建用户界面的控件
Uimenu	创建用户界面的菜单

10.2.2 创建图形对象实例

例 10.1 在 MATLAB 中，创建一个图形对象，并设置简要的对象属性。

step 1 在 MATLAB 的命令窗口中输入下面的命令行：

```
>>x=-4:0.5:4;
>>y=x;
>> [X,Y]=meshgrid(x,y);
>>Z=X.^2-Y.^2;
>>ah=axes('color',[0.8 0.8 0.8],'xtick',[-4 -3 -2 -1 0 1 2 3 4],'ytick',
[-4 -3 -2 -1 0 1 2 3 4]);
>>sh=surface('xdata',X,'ydata',Y,'zdata',Z,'Facecolor',get(ah,'color')
+.1,...
'edgecolor','k','marker','o','markerfacecolor',[0.5 1 .85]);
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 10.2 所示。

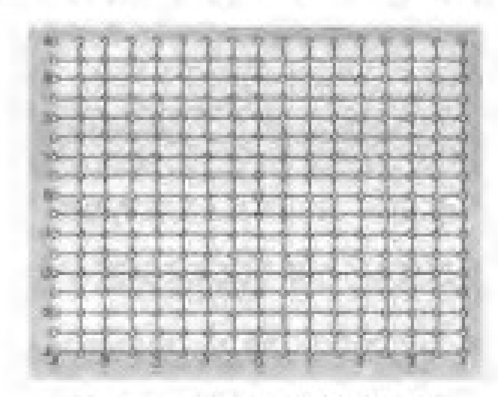


图 10.2 创建图形对象的实例

在上面的程序代码中，通过 axes 命令创建坐标轴对象，同时返回坐标轴对象的句柄 ah，在

该命令行中设置坐标系中 x 和 y 轴的坐标刻度和颜色，然后使用底层命令 surface 来绘制曲面，在其中设置了数据标记、数据点的颜色属性等。在设置数据标记的表面颜色时，使用get函数来获取坐标轴的颜色属性，关于该函数的用法将在后面详细介绍。



从上面的程序结果中可以看出，底层命令 surface 的默认绘制界面是二维图形，而不是三维图形，这个命令和对应的高层命令 surf 有很大的区别。

step 3 在命令窗口中输入下面的命令代码：

```
view(3)
```

step 4 查看图形结果。输入上面的代码后，按“Enter”键，得到的图形如图 10.3 所示。

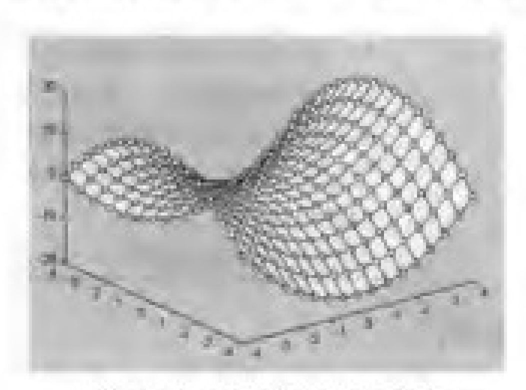


图 10.3　修改视角后的图形

step 5 在上面的图形对象中添加文本对象，在 MATLAB 的命令窗口输入下面的代码：

```
>> text1 = text(...
   'Position',[-36.6 -45.59 186.6],...
   'String','Figure1');
```

step 6 查看图形结果。输入上面的代码后，按“Enter”键，得到的图形如图 10.4 所示。

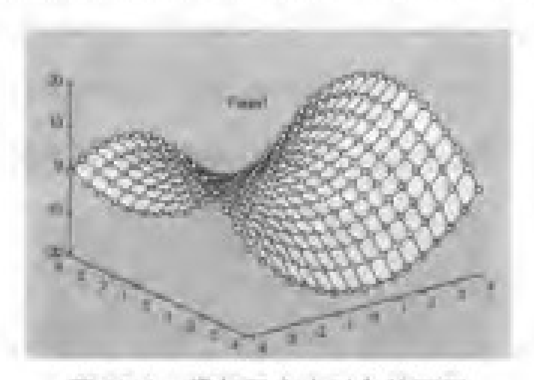


图 10.4　添加了文本对象的图形

上面的对象创建函数还可以使用一个更加简单的调用格式：

```
>> text1 = text(-36.6,-45.59,186.6,'Figure1');
```

该命令和上面步骤中使用的命令是等效的。



上面的命令行调用形式比正常的默认格式要简单很多，但是有时会导致输出效果与正常调用格式有细微的差别，因此如果不太熟悉正常调用格式，不建议使用这种方法。

10.2.3 访问图形对象的句柄

前面已经介绍过，MATLAB会给用户创建的每个图形对象指定一个句柄，所有对象创建函数都能返回对象的句柄。如果需要访问对象的属性，那么最好在创建对象时将对象的句柄赋给一个变量。因为，如果设置图形对象的属性，首先需要知道该对象对应的句柄。在本小节中，将介绍如何访问图形对象的句柄。

在MATLAB中，获取图形对象的句柄有下面几种常见的方法。

◆ 通过图形创建命令获取对象的句柄。在MATLAB中，无论使用高层命令还是底层命令都可以返回图形对象的句柄。例如：

```
>>H_Line1=plot(x,y);
>> text1 = text(-36.6,-45.59,186.6,'Figure1');
```

在上面的程序代码中，H_Line1、text1都是相应图形对象的句柄。

◆ 通过get函数访问图形对象的句柄。如果某图形对象的句柄已知，可以在程序代码中通过get函数来访问该图形对象的句柄。通用的调用格式如下：

```
H_pa=get(H_known,PV)    获取H_known句柄对象的句柄数值
```

◆ 对于用户操作的当前对象，MATLAB提供一些简便的访问方法：

- gcf　返回当前图形窗口（CurrentFigure）的句柄
- gca　返回当前图形窗口中的坐标轴（CurrentAxes）的句柄
- gco　返回最近被鼠标点击的图形对象（CurrentObject）的句柄

◆ 使用对象的"标签"来访问对象句柄。用户可以通过'Tag'属性来给对象设置一个标签，然后通过图形对象的标签来访问对象的句柄。下面的程序代码可以访问相应的图形对象句柄：

```
>>plot(x,y,'Tag','A1')
>>set(gca,'Tag','A1')
```

10.2.4 访问图形句柄实例

下面举例说明如何利用上面介绍的方法来访问图形对象的句柄。

例 10.2　使用高层命令绘制cost在[0, π/2]范围内的图形，然后添加文字注释，最后访问图形对象句柄，修改文字注释的位置。

step 1 在MATLAB的命令窗口输入下面的代码：

```
>> t=0:0.01:2*pi; y=cos(t);
>> plot(t,y) grid on
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图10.5所示。

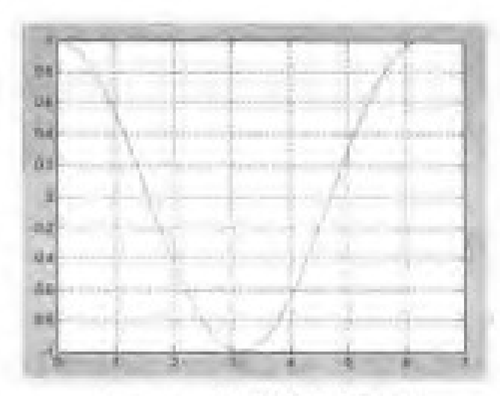


图 10.5　绘制的基础图形

step 3 添加文本注释。在上面步骤完成的图形中添加文本注释，输入下面的代码：

```
>> text(5,0.8,'\fontsize{16} cos(t)','Tag','A1');
```

step 4 查看图形结果。输入上面的代码后，按“Enter”键，得到的图形如图 10.6 所示。

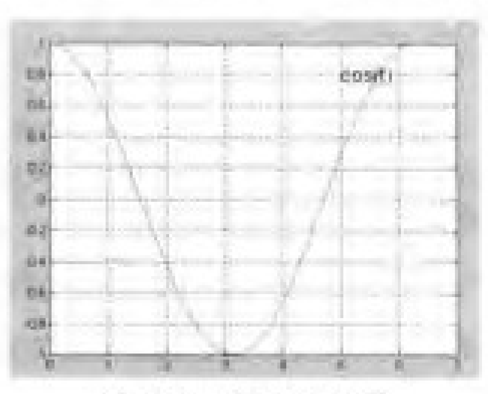


图 10.6　添加文字注释

step 5 获取文本对象的句柄，进行句柄操作。在命令窗口输入下面的代码：

```
>> H=findobj(0,'Tag','A1');
>> set(H,'position',[3 0.8])
```

step 6 查看图形结果。输入上面的代码后，按“Enter”键，得到的图形如图 10.7 所示。

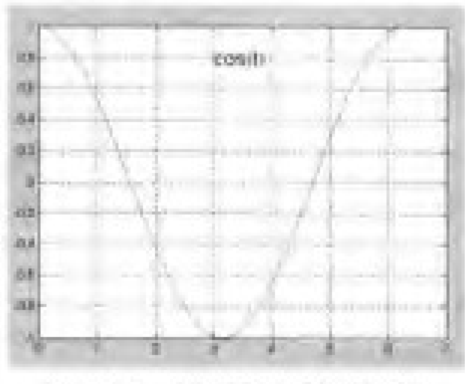


图 10.7　修改文本的位置属性

step 7 查看图形对象的子类型。前面的步骤中已经修改了文本的属性，现在可以看出图形中所有对

象类型，在命令窗口中输入下面的内容：

```
>> H_figure=get(gca,'children');
T=get(H_figure,'type')
T =
    'text'
    'line'
```

可以看出，当前坐标轴的子对象有直线和文本两个对象。上面的代码两次使用了get函数，获取相应的句柄和句柄类型，其中T是元胞数组类型。

10.2.5 使用句柄操作图形对象

在上面小节中，通过使用函数findobj函数查找当前图形对象中的子对象。除了这个函数之外，MATLAB中还提供其他句柄操作函数，使用这些函数用户可以通过句柄操作来修改图形对象的属性。在本小节中，将介绍几个常见的操作函数，帮助读者了解句柄操作的内容。

其中比较常用的函数是copyobj，这个函数的功能是将一个对象从一个父对象复制到另外一个父对象中，其调用格式如下：

```
new_handle = copyobj(h,p)
```

通过该命令创建的新对象与原来的对象唯一的差别在于Parent属性和句柄。用户可以同时将多个对象复制到一个新的父对象中，也可以将一个对象复制到多个父对象中。如果复制的对象包含子对象，MATLAB则会将子对象一起复制过去。

例 10.3 绘制peaks函数的三维图形，然后将其复制到另外一个图形中。

step 1 在MATLAB的命令窗口中输入下面的代码：

```
>> h = surf(peaks);
>> colormap hot
```

step 2 查看图形对象类型。输入代码后，可以查看该图形对象类型，得到的结果如下：

```
>> get(h,'type')
ans =
surface
```

从上面的结果中可以看出，通过peaks函数获得的图形对象是"surface"，而且没有其他的子对象，程序得到的图形如图10.8所示。

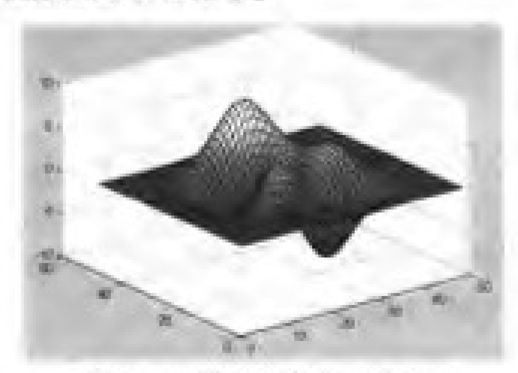


图 10.8 原始的图形对象结果

step 3 复制上面的图形对象。在MATLAB的命令窗口中输入下面的代码：

```
>> figure        % Create a new figure
>>axes          % Create an axes object in the figure
>>new_handle = copyobj(h,gca);
```

step 4 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 10.9 所示。

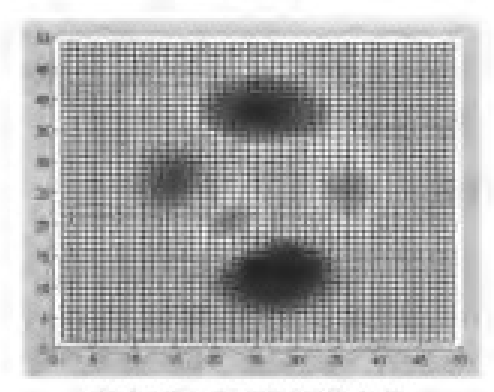


图 10.9　复制的图形对象



从上面的程序结果中可以看出，尽管使用 copyobj 命令复制了图形对象的属性，但是由于底层命令 surface 的默认二维视角，因此复制后的图形会是二维的。

step 5 修改复制后的图形对象的属性，在命令窗口中输入下面的代码：

```
>> colormap cool
>>view(3)
>>grid on
```

step 6 查看图形结果。输入上面的代码后，按"Enter"键，得到的图形如图 10.10 所示。

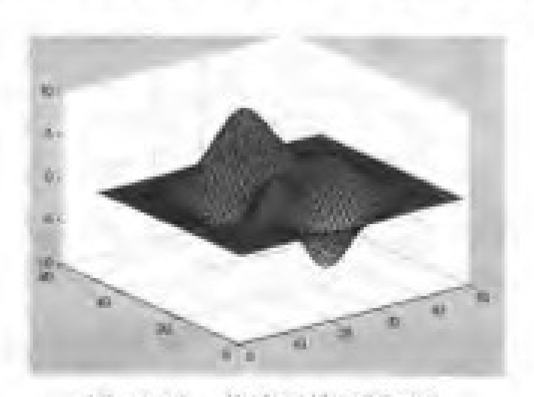


图 10.10　修改后的图形对象

除了使用复制函数之外，MATLAB 还提供 delete 命令来清除图形句柄对象，使用起来非常便捷，而且不会影响其他对象的属性。下面举例说明如何使用 delete 命令清除图形对象。

例 10.4　沿用例 10.2 的例子，在添加文本对象后，使用 delete 命令将其清除。

step 1 在 MATLAB 的命令窗口中输入下面的代码：

```
>> t=0:0.01:2*pi; y=cos(t);
>> plot(t,y)
>>grid on
```

```
>> text(5,0.8,'\fontsize{16} cos(t)','Tag','A1');
```

step 2 查看图形结果。输入上面代码后，按"Enter"键，得到的图形如图 10.11 所示。

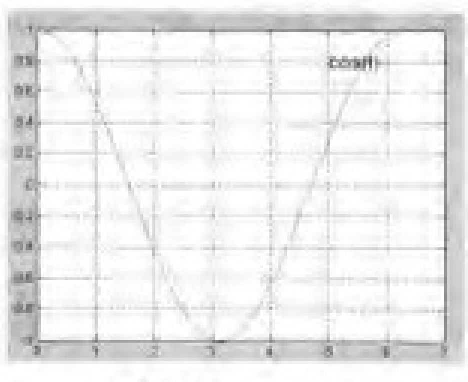


图 10.11　原始图形

step 3 在 MATLAB 的命令窗口中输入下面的代码：

```
>> H=findobj(0,'Tag','A1');
>> delete(H)
```

step 4 查看图形结果。输入上面代码后，按"Enter"键，得到的图形如图 10.12 所示。

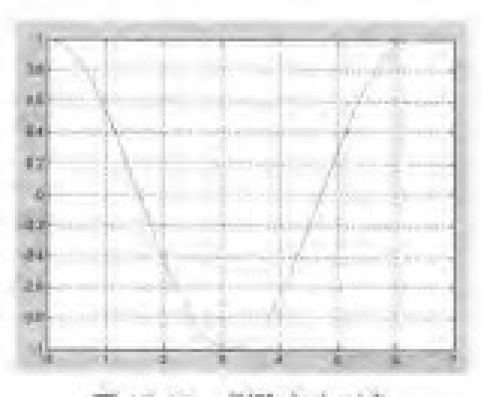


图 10.12　删除文本对象

10.3 图形对象的操作

当通过上面章节中任何一个方法创建图形对象后，对于图形对象的操作主要就是查询、修改或者设置各个图形对象的属性值。对于图形对象，用户访问查询其各个属性值，也可以根据需要设置属性值，最后，可以根据需要设置默认属性值。本节将详细介绍如何修改、访问对象属性和设置默认属性值。

10.3.1 set 命令

在 MATLAB 中，可以使用多种方法来设置创建的图形对象的属性，其中最常用的设置方法包括创建图形时设置属性参数、使用 set 函数设置相关属性、使用结构体数组来定义属性等。

✦ 创建图形对象时设置属性。使用这种方法来设置属性的基本调用格式如下：

```
H_GC=GraphicCommand(......,PN,PV)
```

其中，GraphicCommand代表的是MATLAB中可用的绘图命令，可以是高层命令，也可以是底层命令。(PN，PV)是属性名、属性值构成的属性对，属性对的数目没有任何的限制。例如，下面的命令行就可以在创建对象的时候设置相应的属性：

```
>> plot(x,y,'r*','LineWidth'...
2,'MarkerSize',20)
```

◆ 通过函数set设置图形对象的属性。在MATLAB中，set函数的常用调用格式如下：

```
set(H,'PropertyName',PropertyValue,...)  设置H句柄对象的对应属性的属性值
set(H,'PropertyName')                     显示H句柄对象PropertyName的全部属性值
```

例如，可以通过set函数将图形的y坐标轴移到图形的右侧，对应的代码如下：

```
set(gca,'YAxisLocation','right')
```



set命令实例

例10.5 绘制peaks函数的三维图形，并通过set函数查看各种属性。

step 1 绘制三维图形。在MATLAB的命令窗口中输入下面的代码：

```
>> H_figure=mesh(peaks(40));
```

上面的命令行可以得到peaks函数的曲面图，同时将图形句柄保存在变量H_figure中。为了节省篇幅，这里就不给出具体的三维图形了。

step 2 通过set函数查看用户可以设置的图形属性，得到的结果如下：

```
>>set(H_figure)
ans =
              AlphaData: {}
       AlphaDataMapping: {3x1 cell}
                  CData: {}
           CDataMapping: {2x1 cell}
              EdgeAlpha: {2x1 cell}
              EdgeColor: {3x1 cell}
              EraseMode: {4x1 cell}
              FaceAlpha: {3x1 cell}
              FaceColor: {4x1 cell}
              LineStyle: {5x1 cell}
              LineWidth: {}
                 Marker: {14x1 cell}
        MarkerEdgeColor: {3x1 cell}
        MarkerFaceColor: {3x1 cell}
             MarkerSize: {}
              MeshStyle: {3x1 cell}
                  XData: {}
                  YData: {}
                  ZData: {}
           FaceLighting: {4x1 cell}
           EdgeLighting: {4x1 cell}
```

```
     BackFaceLighting: {3x1 cell}
      AmbientStrength: {}
..................................//限于篇幅,省略了部分属性数值
           BusyAction: {2x1 cell}
     HandleVisibility: {3x1 cell}
              HitTest: {2x1 cell}
        Interruptible: {2x1 cell}
             Selected: {2x1 cell}
   SelectionHighlight: {2x1 cell}
                  Tag: {}
        UIContextMenu: {}
             UserData: {}
              Visible: {2x1 cell}
               Parent: {}
          DisplayName: {}
            XDataMode: {2x1 cell}
          XDataSource: {}
            YDataMode: {2x1 cell}
          YDataSource: {}
            CDataMode: {2x1 cell}
          CDataSource: {}
          ZDataSource: {}
```

在上面的结果中，如果某项属性对应的属性值为空，例如"LineWidth: {}"，表示不能对该图形对象设置LineWidth属性。如果对应的属性值不为空，则可以在对应的属性值列表中设置相应的属性。

step 3 通过set函数查看图形的"Marker"属性列表，得到的结果如下：

```
>> set(H_figure,'Marker')
[ + | o | * | . | x | square | diamond | v | ^ | > | < | pentagram |
hexagram | {none} ]
```

从上面的结果可以看出，可以从上面的"Marker"属性的14个选项中选择任何一个选项来设置图形标记的属性。

step 4 通过set函数设置图形标记的属性，相关的代码如下：

```
>>set(H_figure,'Marker','o')
```

step 5 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图10.13所示。

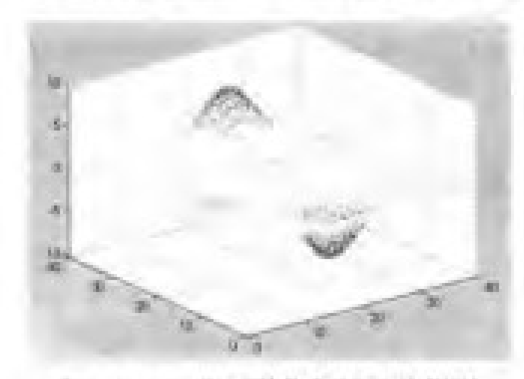


图 10.13　设置数据标记后的图形



使用结构体设置属性

除了上面的方法之外，在MATLAB中还可以使用结构体数组来直接设置图表对象的属性。例如，可以定义一个结构体来设置图表对象的属性：

```
props.FaceColor= 'texture';
props.EdgeColor = 'none';
props.FaceLighting = 'phong';
```

然后通过set函数选用上面的结构体props，设置图形对象的属性，相应的代码如下：

```
set(gcf,props)
```

相当于对当前图形窗口设置结构体props所定义的对应的属性。下面将介绍几个综合设置图形对象属性的例子。

例 10.6 在MATLAB中绘制克莱因瓶（Klein bottle）。

step 1 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
n = 12;
a = input('Enter the diameter of the small tube:');  % 小径的直径
c = input('Enter the diameter of the bulb:');        %大径的直径
t1 = pi/4 : pi/n : 5*pi/4;      % 沿着管子的参数
t2 = 5*pi/4 : pi/n : 9*pi/4;    % 环绕管子的角度
u  = pi/2 : pi/n : 5*pi/2;
[X,Z1] = meshgrid(t1,u);
[Y,Z2] = meshgrid(t2,u);

% 绘制图形的把手
len = sqrt(sin(X).^2 + cos(2*X).^2);
x1 = c*ones(size(X)).*(cos(X).*sin(X) ...
    - 0.5*ones(size(X))+a*sin(Z1).*sin(X)./len);
y1 = a*c*cos(Z1).*ones(size(X));
z1 = ones(size(X)).*cos(X) + a*c*sin(Z1).*cos(2*X)./len;
handleHndl=surf(x1,y1,z1,X);
set(handleHndl,'EdgeColor',[.5 .5 .5]);
hold on;

% 绘制径
r = sin(Y) .* cos(Y) - (a + 1/2) * ones(size(Y));
x2 = c * sin(Z2) .* r;
y2 = - c * cos(Z2) .* r;
z2 = ones(size(Y)) .* cos(Y);
bulbHndl=surf(x2,y2,z2,Y);
set(bulbHndl,'EdgeColor',[.5 .5 .5])
colormap(hsv);
axis vis3d
view(-37,30);
axis off
light('Position',[2 -4 5])
light
hold off
```



克莱因瓶是指一种既没有内表面也没有外表面的单侧拓扑面，它是通过把逐渐变细的管子的小的开端插入管壁并使它与大的开端相连而形成的。克莱因瓶主要参数是球体和导管的半径，也就是上面程序代码中的变量a和c的数值。

step 2 将上面的程序代码保存为文件"klein.m"，返回MATLAB的命令窗口中，输入"klein"，得到的结果如下：

```
>> klein
Enter the diameter of the small tube:0.3
Enter the diameter of the bulb:0.7
```

step 3 查看图形结果。输入上面的参数数值后，按"Enter"键，得到的图形如图10.14所示。



图 10.14 绘制得到的克莱因瓶

step 4 修改两个半径参数的数值，重新绘制克莱因瓶，得到的结果如下

```
>> klein
Enter the diameter of the small tube:0.1
Enter the diameter of the bulb:0.9
```

step 5 查看图形结果。输入上面的参数值后，按"Enter"键，得到的图形如图10.15所示。



图 10.15 修改曲面参数绘制图形



在上面的程序代码中，分别使用高层和底层命令来设置图形的各种属性。可以看出，并不复杂的程序可以绘制出比较复杂的图形。

例 10.7 在MATLAB中绘制三维的地球图形。

step 1 单击命令窗口工具栏中的按钮，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
%加载系统自带的数据文件
load topo;
%绘制球体
[x,y,z] = sphere(50);
cla reset
axis square off
%设置结构体props的属性
props.AmbientStrength = 0.1;
props.DiffuseStrength = 1;
props.SpecularColorReflectance = .5;
props.SpecularExponent = 20;
props.SpecularStrength = 1;
props.FaceColor= 'texture';
props.EdgeColor = 'none';
props.FaceLighting = 'phong';
%定义图形的数据
props.Cdata = topo;
surface(x,y,z,props);
%设置光照的属性
light('position',[-1 0 1]);
light('position',[-1.5 0.5 -0.5], 'color', [.6 .2 .2]);
%设置图形的视角
view(3)
```

step 2 将上面的程序代码保存为文件"three_globe.m"，返回MATLAB的命令窗口中，输入"three globe"，得到的结果如图10.16所示。



图10.16　程序绘制得到的图形

在上面的程序代码中，通过代码"load topo"，加载topo.mat文件中的数据程序。topo是MATLAB的自带数据文件，默认的保存路径为.....\MATLAB7.0\toolbox\matlab\general。



说明　在上面的程序代码中，首先定义结构体props，然后在绘图命令surface中引用该结构体，通过这种方法来设置图形对象的属性。

10.3.4 查询图形对象的属性

在MATLAB中，可以使用get函数来查询图形对象的属性值。其常见的调用格式如下：

- get(h)　获取h句柄对象所有属性的当前值；
- get(h,'PropertyName')　获取h句柄对象由PropertyName所指定属性的当前值。

和前面介绍的set函数类似，get函数可以获取各种层次的对象属性值。

例 10.8 创建简单的图形对象，然后使用 get 函数获取各种对象的属性。

step 1 创建简单的图形对象。在 MATLAB 的命令窗口中输入下面的代码：

```
>>patch;surface;text;line
```



上面简单的程序代码创建了 4 个图形对象，由于这些对象都没有使用参数，MATLAB 会采用各个对象的默认属性来绘制对象。同时，在创建这些对象之前，并没有创建图形窗口或者坐标轴对象，MATLAB 也会按照默认属性来创建这些对象。

step 2 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图 10.17 所示。

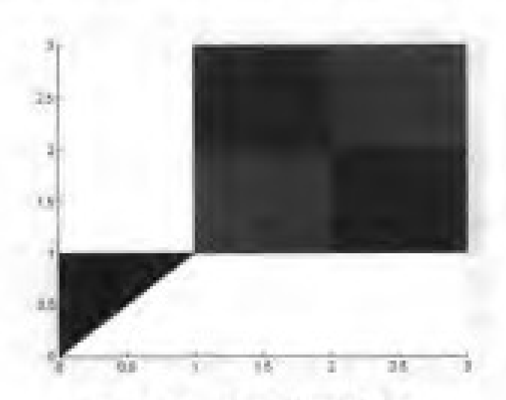


图 10.17 创建的图形对象

step 3 查看该图形对象的所有属性值。在 MATLAB 的命令窗口中输入"get(gca)"，得到用户绘制的图形对象的所有属性值，结果如下：

```
>> get(gca)
    ActivePositionProperty = outerposition
    ALim = [ 0.1 10]
    ALimMode = auto
    AmbientLightColor = [1 1 1]
    Box = off
    CameraPosition = [ 1.5 1.5 9.16025]
    CameraPositionMode = auto
    CameraTarget = [ 1.5 1.5 0.5]
    CameraTargetMode = auto
    CameraUpVector = [ 0 1 0]
    CameraUpVectorMode = auto
    CameraViewAngle = [ 6.60861]
    CameraViewAngleMode = auto
    CLim = [ 0 1]
    CLimMode = auto
    Color = [1 1 1]
    CurrentPoint = [  (2 by 3) double array]
    ColorOrder = [  (7 by 3) double array]
    ........................................//限于篇幅，省略了部分属性数值
    Interruptible = on
    Parent = [ 1]
    Selected = off
    SelectionHighlight = on
    Tag =
```

```
    Type = axes
    UIContextMenu = []
    UserData = []
    Visible = on
```

在上面的属性数值列表中，许多属性选项都列出了具体的数值，例如"CameraPosition = [1.5 1.5 9.16025]"，但是，也有一些属性列表只是列出了数值的维度，例如"ColorOrder = [(7 by 3) double array]"。如果需要了解对应属性的数值，则需要调用相应的格式，查看具体数值。

step 4 查看该图形对象的"ColorOrder"属性值。在MATLAB的命令窗口中输入"get(gca,'ColorOrder')"，得到的结果如下：

```
>> get(gca,'ColorOrder')
ans =
         0         0    1.0000
         0    0.5000         0
    1.0000         0         0
         0    0.7500    0.7500
    0.7500         0    0.7500
    0.7500    0.7500         0
    0.2500    0.2500    0.2500
```

step 4 定义结构体，查询结构体中指代的图形对象属性。在MATLAB的命令窗口中输入代码：

```
>> props = {'HandleVisibility', 'Interruptible';
'SelectionHighlight', 'Type'};
>>output = get(get(gca,'Children'),props);
```

step 6 查看查询结果。在MATLAB的命令窗口中输入"output"，按"Enter"键，查看属性查询结果，得到的结果如下：

```
>> output
output =
    'on'    'on'    'on'    'line'
    'on'    'on'    'on'    'text'
    'on'    'on'    'on'    'surface'
    'on'    'on'    'on'    'patch'
```

查看图形对象的默认属性

在MATLAB中，如果需要了解图形对象的所有系统默认属性，可以通过命令代码get(0,'factory')获得，得到的结果如下：

```
           factoryFigureAlphamap: [1x64 double]
        factoryFigureBackingStore: 'on'
          factoryFigureBusyAction: 'queue'
       factoryFigureButtonDownFcn: ''
           factoryFigureClipping: 'on'
    factoryFigureCloseRequestFcn: 'closereq'
              factoryFigureColor: [0 0 0]
           factoryFigureColormap: [64x3 double]
          factoryFigureCreateFcn: ''
```

```
            factoryFigureDeleteFcn: ''
          factoryFigureDockControls: 'on'
....................................//限于篇幅,省略了部分属性数值
               factoryRootHitTest: 'on'
          factoryRootInterruptible: 'on'
         factoryRootRecursionLimit: 2.1475e+009
    factoryRootScreenPixelsPerInch: 96
     factoryRootSelectionHighlight: 'on'
      factoryRootShowHiddenHandles: 'off'
                   factoryRootTag: ''
              factoryRootUserData: []
               factoryRootVisible: 'on'
```

从上面的结果中可以看出，系统默认的属性一般的标号为FactoryPropertyName，后面显示的就是默认属性数值。

如果希望了解某种属性的具体数值，可以使用下面的类似代码获得：

```
>> get(0,'factoryFigurePaperPosition')
ans =
    0.2500    2.5000    8.0000    6.0000
```



对于所有图形对象的属性，MATLAB 系统都会有一个默认的系统属性值。

在 MATLAB 中，除了可以查询系统的默认属性值之外，还可以根据需要自定义各种图形对象的属性默认值。在介绍如何设置对象的默认属性值之前，将首先介绍 MATLAB 选用属性值的原理。

在MATLAB中进行绘图或者需要了解对象属性值时，MATLAB会从当前对象开始，在继承表中向上搜索，直到找到系统定义或者用户自行设置的默认属性值。因此，定义的默认值越靠近 root 对象，MATLAB 搜索的范围也就越广。例如，在 root 级定义 line 对象的默认值，MATLAB 会对所有的 line 对象使用这个默认值，如果在axes级中定义 line对象的默认值，MATLAB 会对该图形对象中的 line 对象使用该默认值。



如果在多级目录中定义了对象的默认值，那么与该对象最近的父对象级中定义的默认值将被使用，因为该默认值是 MATLAB 中最先找到的默认值。

还需要提醒读者的是，用户设置的默认值只对设置完成后创建的对象有效，对于之前创建的图形对象无效。设置属性的默认值使用前面小节的set 命令，只是在设定之前需要创建一个以 Default 开头、然后跟着对象类型、最后是对象属性的字符串。例如，用户希望在当前的图形窗口级设置 line 对象的 LineWidth（线宽）默认属性值为 2.5 点宽，可以使用下面的程序代码：

```
set(gcf,'DefaultLineLineWidth',2.5)
```

10.3.6 设置不同级别的属性

例 10.9 对 MATLAB 设置图形对象中不同级别的默认属性，并查看设置效果。

step 1 在 MATLAB 的命令窗口中输入下面的代码：

```
>> set(0,'DefaultSurfaceMarker','o');
>>h=surface(sphere(30));
```

```
>>view(3);
>> grid
```

step 2 查看图形结果。输入程序代码后，按 "Enter" 键，得到的图形如图 10.18 所示。

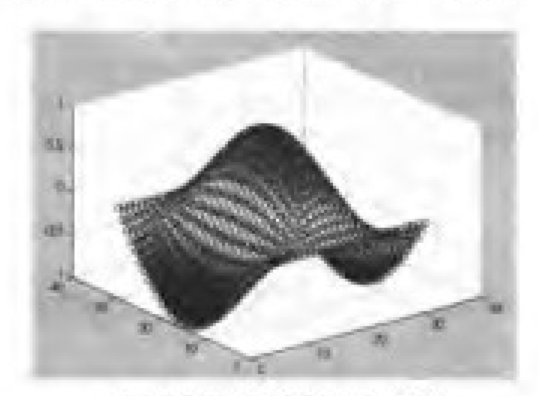


图 10.18　得到的初始图形

step 3 重新设置图形标记图案的默认值。在 MATLAB 的命令窗口中输入下面的代码：

```
>> set(gcf,'DefaultSurfaceMarker','*');
>> set(h,'Marker','default')
```

step 4 查看图形结果。输入程序代码后，按 "Enter" 键，得到的图形如图 10.19 所示。

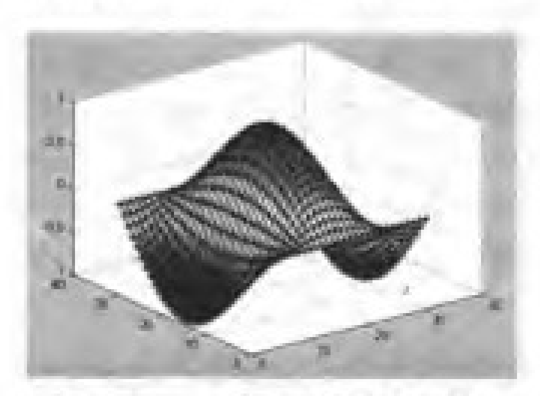


图 10.19　修改图形标记的默认值

step 5 打开新的图形窗口，绘制三维图形。在命令窗口中输入下面的代码：

```
>> figure
>> h=surface(sphere(40));
>>view(3);
>> grid
```

step 6 查看图形结果。输入程序代码后，按 "Enter" 键，得到的图形如图 10.20 所示。

从上面的代码中可以看出，设置默认属性数值的级别会影响图形绘制的结果。在第 2 个步骤中，设置的默认图形标记属性的级别为当前图形窗口（gcf），因此其默认属性值只适用在图形窗口 Figure1 中。因此，在步骤 3 中绘制新的图形对象时，使用的是步骤 1 中定义在根目录中的默认属性值。

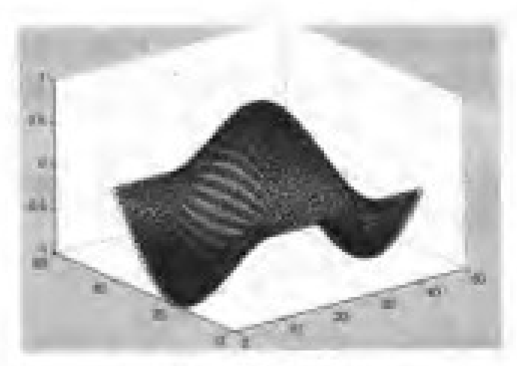


图 10.20 重新绘制图形



从上面的程序代码中可以看出，在不同级别中定义默认属性值，其相应的使用范围和方法都会有很大的区别，需要谨慎地设置不同级别的默认数值。

10.3.7 设置图形对象的默认属性

在 MATLAB 中，除了可以设置不同使用范围和级别的默认属性之外，也可以根据需要删除用户设定的默认属性。如果需要删除设置的图形标记的默认属性，可以使用下面的程序代码：

```
set(0,'DefaultSurfaceMarker','remove');
```

在输入上面的代码后，查看系统中图形标记的默认属性，得到的结果如下：

```
>> get(0,'DefaultSurfaceMarker')
ans =
none
```

从上面的程序代码中可以看出，在删除用户设置的默认属性后，MATLAB 会返回系统设定的默认属性 none，表明已经删除用户设定的默认属性。除了可以删除用户设定的默认属性之外，也可以不使用用户定义的默认属性，沿用上面的例子，在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> set(0,'DefaultSurfaceMarker','o');
>>h=surface(sphere(30));
>>view(3);
>> set(h,'Marker','factory')
>> grid
```

当输入上面的代码后，按 "Enter" 键，得到的结果如图 10.21 所示。

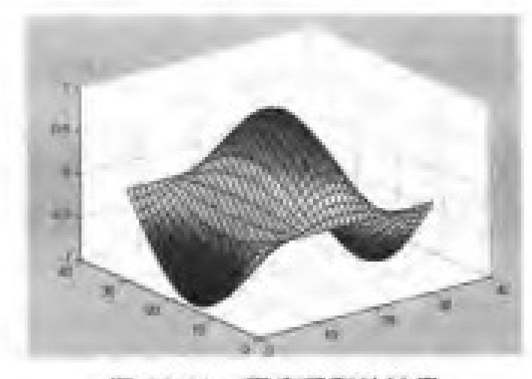


图 10.21 程序得到的结果

从上面的程序代码中可以看出，尽管设置了图形标记的默认图案是"o"，但是在程序代码中将图形的标记设置为"factory"（系统默认）的属性值，因此，得到的图形中图形标记还是默认的none，而不是用户设置的"o"。

例 10.10　设置当前坐标轴（gca）的默认属性，然后使用该默认属性来绘制图形。

step 1 在 MATLAB 的命令窗口中输入下面的代码：

```
>> whitebg('w');
>> set(0,'DefaultAxesColorOrder',[1 0.5 0],'DefaultAxesLineStyleOrder',
'-|:|--|-.',...
'DefaultAxesYGrid','on','DefaultLineLineWidth',2);
>>z=sphere(50);
>>plot(1:51,z(3:6,:))
```

step 2 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图 10.22 所示。

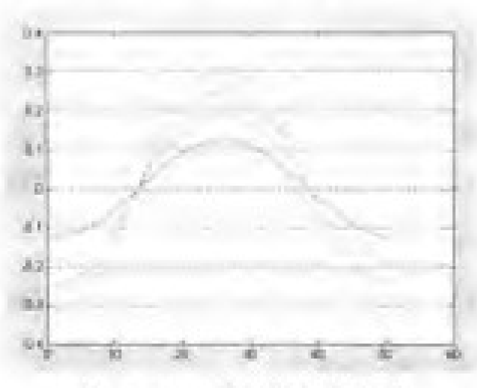


图 10.22　绘制得到的图形



在上面的程序代码中，设置了默认图形颜色为橙色，同时设置每个图形中添加Y轴的网格线，直线的线宽为2等属性。这些属性都是用户自行设置的，修改了系统默认的所有相关属性。只要用户在此根目录下绘制图形，都将沿用该属性。

step 3 在新窗口中绘制图形。在 MATLAB 的命令窗口中输入下面的代码：

```
>> figure
>>t=0:0.01:2*pi;
>>y1=sin(t);
>>y2=cos(t);
>>plot(t,y1,t,y2))
```

step 4 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图 10.23 所示。



从上面的程序代码中可以看出，如果不删除设置的默认属性数值，在该root对象下绘制的所有图形对象都将沿用该属性体系。

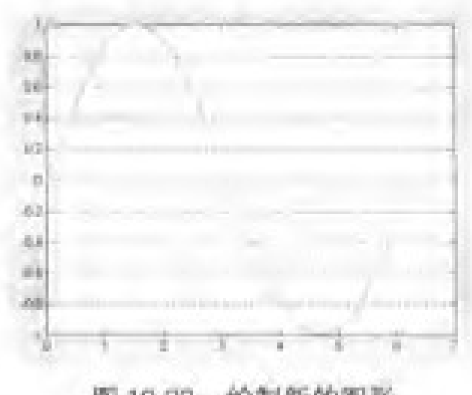


图 10.23 绘制新的图形

10.4 高层绘图命令

在详细介绍图形中的各个对象属性之前，首先介绍高层绘图命令文件的组成情况，这样可以帮助读者更好地理解高层绘图命令。同时还可以帮助读者理解底层绘图命令如何设置图形父对象的属性。

10.4.1 NextPlot 属性

当使用底层命令创建线、面、方块等子对象时，就会涉及如何在 MATLAB 系统中为这些对象准备图、轴的"父"对象的问题。在 MATLAB 中，这个问题相当于如何设置 Axes、Figure 的"NextPlot"属性的问题。

Figure 对象的"NextPlot"属性如下。

- Add：表示在当前设置下，允许添加子对象。其对应的高层命令为 hold on。
- Replacechildren：表示在当前设置下，清除所有子对象。对应的高层命令为 clf。
- Replace：表示清除所有子对象，重新设置为默认值，对应的高层命令为 clf reset。

Axes 对象的"NextPlot"属性如下。

- Add：表示在当前设置下，允许添加子对象。其对应的高层命令为 hold on。
- Replacechildren：表示在当前设置下，清除所有子对象。对应的高层命令为 cla。
- Replace：表示清除所有子对象，重新设置为默认值，对应的高层命令为 cla reset。



在 MATLAB 中，Figure 的"NextPlot"默认属性是"Add"，Axes 的"NextPlot"默认属性是"Replace"。另外，两个对象的重设命令"reset"不影响 Position、Units 的属性。

10.4.2 Newplot 命令

在 MATLAB 中，为了方便用户开发图形文件，专门设计了 newplot 命令，该命令将自动执行对当前图、轴对象的"NextPlot"属性进行检查，并完成相应的设置，具体功能如下。

检查并设置当前图形窗的"NextPlot"属性。

✦ 如果结果是 ReplaceChildren，则清除图形中全部的子对象；
✦ 如果结果是 Replace，则清除图形中全部的子对象，并将图的对象属性设置为系统的默认值；
✦ 如果结果是 Add，则保留当前图形窗口中所有子对象和所有属性不变。

检查并设置当前图形轴的“NextPlot”属性：

✦ 如果结果是 ReplaceChildren，则清除图形轴中全部的子对象；
✦ 如果结果是 Replace，则清除图形轴中全部的子对象，并将轴的对象属性设置为系统的默认值；
✦ 如果结果是 Add，则保留当前图形轴中所有子对象和所有属性不变。

高层绘图文件的构成

前面已经列出MATLAB所有相关的底层绘图命令，而且已经向读者介绍过。MATLAB的高层绘图函数都建立在底层绘图命令之上，下面分析MATLAB的mesh.m文件，说明mesh命令是如何建立在surface命令之上的。

例 10.11 分析高层绘图命令 mesh 的文件构成。

在 MATLAB 的默认路径...\MATLAB7.0\toolbox\matlab\general 中，可以查看 mesh.m 文件的程序代码：

```
[ mesh.m]
function h = mesh(varargin)
[ v6,args] = usev6plotapi(varargin{ :} );
[ cax,args,nargs] = axescheck(args{ :} );
user_view = 0;
cax = newplot(cax);
hparent = get(cax,'parent');
fc = get(cax,'color');
if strcmpi(fc,'none')
  if isprop(hparent,'Color')
   fc = get(hparent,'Color');
  elseif isprop(hparent,'BackgroundColor')
   fc = get(hparent,'BackgroundColor');
  end
end
[ reg, prop] =parseparams(args);
nargs=length(reg);
error(nargchk(1,4,nargs));
if rem(length(prop),2)~=0,
   error('Property value pairs expected.')
end
   if nargs == 1
   x=reg{ 1} ;
   if v6
       hh = surface(x,'FaceColor',fc,'EdgeColor','flat', ...
           'FaceLighting','none','EdgeLighting','flat','parent',cax);
   else
       hh = graph3d.surfaceplot(x,'FaceColor',fc,'EdgeColor','flat', ...
           'FaceLighting','none','EdgeLighting','flat','parent',cax);
   end
...................................//限于篇幅，省略了部分程序代码
```

```
elseif nargs == 4
   [x,y,z,c]=deal(reg{1:4});
   if v6
        hh = surface(x,y,z,c,'FaceColor',fc,'EdgeColor','flat', ...
                   'FaceLighting','none','EdgeLighting','flat','parent',cax);
   else
       hh = graph3d.surfaceplot(x,y,z,c,'FaceColor',fc,'EdgeColor','flat', ...
                   'FaceLighting','none','EdgeLighting','flat','parent',cax);
   end
end
if ~isempty(prop),
   set(hh,prop{:})
end
if ~ishold(cax) && ~user_view
   view(cax,3); grid(cax,'on');
end
if nargout == 1
   h = double(hh);
end'
```

从上面的程序代码中，可以看到典型的高层绘图文件的构成：

- 调用命令 newplot，检查并设置图和轴的 NextPlot 属性，然后返回目标轴的句柄；
- 引用 newplot 返回的坐标轴句柄，或者修改坐标轴的属性，或者访问坐标轴的属性；
- 调用图形对象创建命令，创建图形。

坐标轴对象

在图形对象的树形结构中，坐标轴对象（Axes）发挥着重要的作用，其具体的属性有 80 多个。本节将介绍坐标轴对象中一些常见和重要的属性，将会帮助读者更好地设置图形对象的属性。

坐标轴的几何属性

与前面介绍的图形对象类似，轴位框的几何属性主要由 Position、Units 来指定。在 MATLAB 中，轴位框（Axes Position Rectangle）的含义不同于坐标框。对于二维图形，两者指定的是同一个面积；对于三维图形，轴位框指定图形所占用的最大平面面积，而不是三维坐标轴。由于屏幕上没有显示轴位框线，因此这个概念初学者经常会混淆。

在 MATLAB 中，关于坐标轴的"单位"（Units）的默认属性为"normalized"，可以使用 get 函数得到，相应的代码如下：

```
>> get(gca,'units')
ans =
normalized
```

该属性的含义是轴对象使用"归一化"单位。在这种单位下，图形对象窗口的几何属性值总是[0，0，1，1]，其他对象的单位则使用相对单位。



采用归一化单位的结果就是表明坐标轴和图形窗口的相对大小，当对图形窗进行缩放时，坐标轴也会随之缩放，保持相对大小不变。如果采用其他单位设置，则就不能保持这种特性。

在MATLAB的高层命令中，可以使用subplot在同一个图形窗中创建多个坐标轴对象，而且还可以产生任意大小的轴位框。但是各个轴位框不能重叠，这个重叠问题必须使用底层命令来解决，涉及到的坐标轴属性就是Position和Units，下面举例详细介绍。

例 10.12 在MATLAB中，在同一个图形窗口中绘制三个函数的图形。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function [ ax,hlines] = plotyyy(x1,y1,x2,y2,x3,y3,ylabels)
%PLOTYYY - Extends plotyy to include a third y-axis
%
%Syntax:  [ ax,hlines] = plotyyy(x1,y1,x2,y2,x3,y3,ylabels)
%
%Inputs: x1,y1 are the xdata and ydata for the first axes' line
%        x2,y2 are the xdata and ydata for the second axes' line
%        x3,y3 are the xdata and ydata for the third axes' line
%        ylabels is a 3x1 cell array containing the ylabel strings
%
%Outputs: ax -     3x1 double array containing the axes' handles
%         hlines - 3x1 double array containing the lines' handles
if nargin==6
   ylabels{1}=' '; ylabels{2}=' '; ylabels{3}=' ';
end
figure('units','normalized',...
       'DefaultAxesXMinorTick','on','DefaultAxesYminorTick','on');
[ ax,hlines(1),hlines(2)] = plotyy(x1,y1,x2,y2);
cfig = get(gcf,'color');
pos = [ 0.1  0.1  0.7  0.8];
offset = pos(3)/5.5;
pos(3) = pos(3) - offset/2;
set(ax,'position',pos);
pos3=[ pos(1) pos(2) pos(3)+offset pos(4)];
limx1=get(ax(1),'xlim');
limx3=[ limx1(1)   limx1(1) + 1.2*(limx1(2)-limx1(1))];
ax(3)=axes('Position',pos3,'box','off',...
   'Color','none','XColor','k','YColor','r',...
   'xtick',[],'xlim',limx3,'yaxislocation','right');
hlines(3) = line(x3,y3,'Color','r','Parent',ax(3));
limy3=get(ax(3),'YLim');
line([ limx1(2) limx3(2)],[ limy3(1) limy3(1)],...
   'Color',cfig,'Parent',ax(3),'Clipping','off');
axes(ax(2))
set(get(ax(1),'ylabel'),'string',ylabels{1})
set(get(ax(2),'ylabel'),'string',ylabels{2})
set(get(ax(3),'ylabel'),'string',ylabels{3})
```

step 2 单击M文件编辑器中的“保存”按钮，将上面的程序代码保存为“plotyyy”，返回MATLAB的命令窗口中输入下面的代码：

```
>> x=0:0.01:10;
y1=x.^2;
y2=x;
y3=exp(x);
h=plotyyy(x,y1,x,y2,x,y3);
```

step 3 查看图形结果。输入程序代码后，按 "Enter" 键，得到的图形如图 10.24 所示。

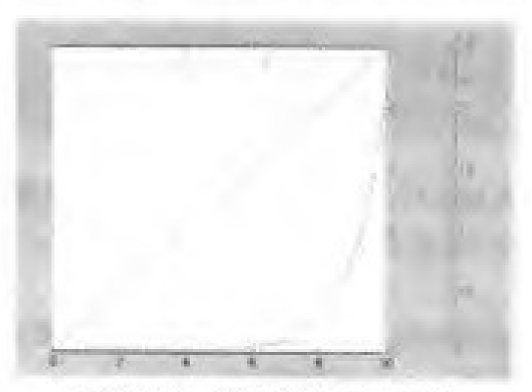


图 10.24　程序所绘制的图形

step 4 重新绘制图形。在 MATLAB 的命令窗口中输入下面的代码：

```
>>figure;
>> t1=0:0.01:2*pi;
>>t2=0.5*pi:0.01:2.5*pi;
>>t3=pi:0.01:1.5*pi;
>>y1=sin(t1);
>> y2=t2.*cos(t2);
>>y3=sqrt(t3).*exp(t3);
>> label=('sin(t)';'t*cos(t)';'sqrt(t)*exp(t)');
>> [a,h] =plotyyy(t1,y1,t2,y2,t3,y3,label);
>> set(h,'LineWidth',2)
>> grid
```

step 5 查看图形结果。输入程序代码后，按 "Enter" 键，得到的图形如图 10.25 所示。

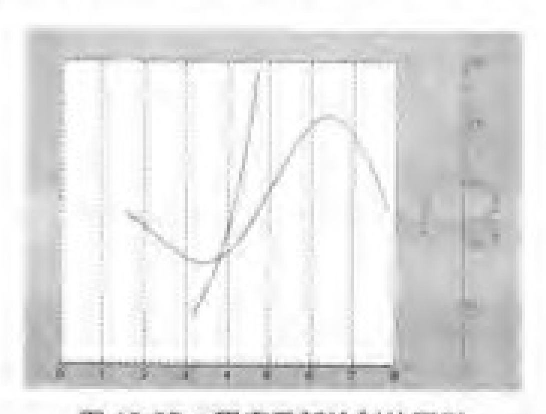


图 10.25　程序重新绘制的图形



在上面的程序代码中，使用变量 pos 来设置三个函数图形所在的轴位框，由于第三个函数的轴位框采用了 offset 变量，因此第三个函数所在的坐标轴向右偏移。

10.5.2 坐标轴的刻度属性

在MATLAB中，没有现成的高层命令来设置图形的坐标轴刻度，如果希望设置图形坐标轴的刻度，则需要调用相应的底层命令：

- set(gca,'Xtick',xs,'Ytick',ys) 设置二维坐标刻度；
- set(gca,'Xtick',xs,'Ytick',ys,'Ztick',zs) 设置三维坐标刻度。



在上面的命令中，xs、ys和zs可以是任何合法的实数向量，它们分别设置x、y和z轴的刻度。

在MATLAB中，除了可以设置坐标轴的刻度属性之外，还可以设置坐标轴的刻度模式，可以选择auto、manual两种刻度模式。由于比较简单，这里就不展开解释了，感兴趣的读者请查阅相应的帮助文件。

例 10.13 在MATLAB中，绘制图形并自行设置坐标轴刻度的属性。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>>x2=0:0.02:16;
>>y2 = 1 ./ ((x2-3).^2 + 1) + 1 ./ ((x2-9).^2 + 4) + 5;
>>plot(x2,y2,'r-','LineWidth',2)
>>set(gca,'Xtick',[0:1:16],'Ytick',[5.1118,5.2770,6.0250],...
'YtickLabel',{'5.1118 ';'5.2770';'最大值'});
>>grid
>>hold on;
>>set(gca,'Ygrid','on')
```

step 2 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图10.26所示。

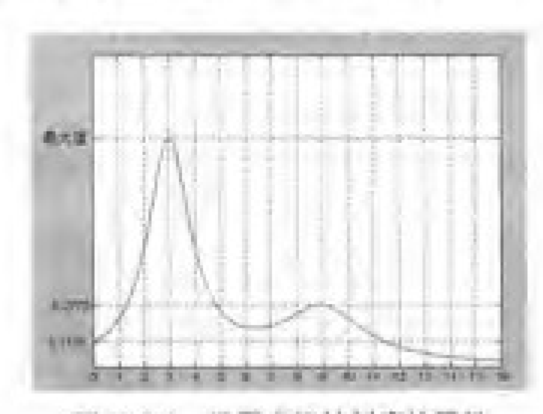


图 10.26 设置坐标轴刻度的属性



在MATLAB中，坐标轴刻度位置的属性必须是一维数值数组，可以是等分数组也可以不是等分数组。而且，坐标轴刻度属性的元素数目必须和刻度位置数组的元素相等。当用户对坐标轴设置属性值时，就不需要设置取值模式，MATLAB会自动决定取值模式。

10.5.3 坐标轴的照相机属性

在MATLAB中，可以设置坐标轴的照相机属性。这些属性都是以坐标轴的属性为基础的，坐标轴

的属性可以控制照相机的位置和角度。一般来讲，用户可以使用相应的命令来直接访问坐标轴的照相机属性，表 10.2 列出了其属性的设置和含义。

表 10.2 照相机属性列表

属性	含义
CameraPosition	照相机的位置:[x,y,z]
CameraPositionMode	照相机位置属性的取值模式
CameraTarget	照相机的目标:[x,y,z]
CameraTargetMode	照相机目标属性的取值模式
CameraUpVector	照相机正位向量:[x,y,z]
CameraUpVectorMode	照相机正位向量的取值模式
CameraViewAngle	照相机的视角
CameraViewAngleMode	照相机视角的取值模式
Projection	照相机的投影方式

上面关于照相机的各个属性的物理含义如图 10.27 所示。

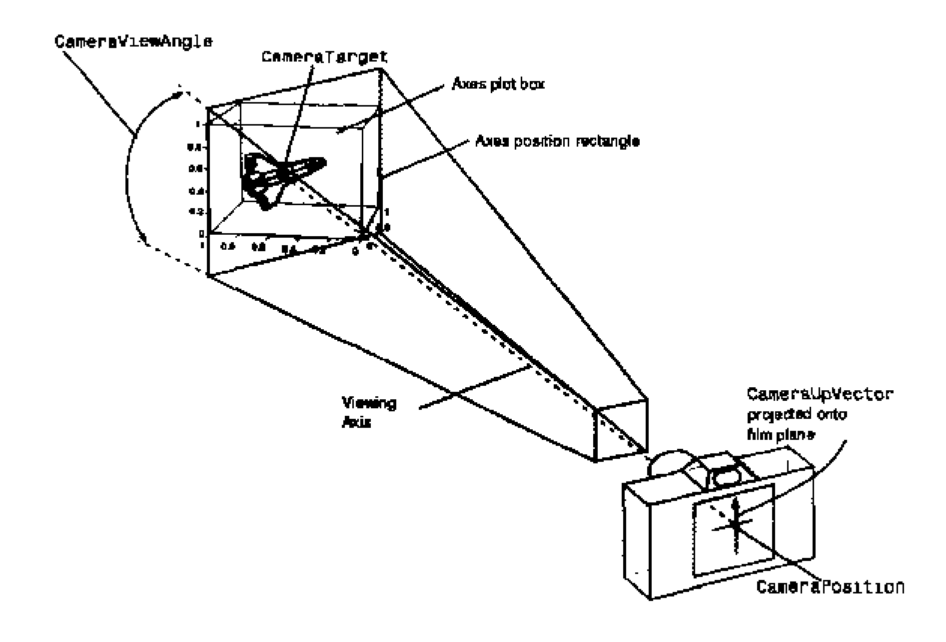


图 10.27 照相机属性的物理意义

在 MATLAB 中，照相机的 CameraTarget 属性在默认情况下的自动取值为坐标框的中心；CameraViewAngle 在自动取值情况下取最小角，使场最张满整个轴位框；同时，照相机的 Projection 属性值为正视投影。

例 10.14 在 MATLAB 中，使用相应的程序代码进行数据可视化，并且选用合适的照相机属性。

step 1 单击 MATLAB 命令窗口工具栏中的 □ 按钮，打开 M 文件编辑器。在 M 文件编辑器中输入下面的程序代码：

```
colordef(gcf,'black')

cla
load wind
spd = sqrt(u.*u + v.*v + w.*w);
p = patch(isosurface(x,y,z,spd, 40));
```

```
isonormals(x,y,z,spd, p)
set(p, 'FaceColor', 'red', 'EdgeColor', 'none');
p2 = patch(isocaps(x,y,z,spd, 40));
set(p2, 'FaceColor', 'interp', 'EdgeColor', 'none')
daspect([1 1 1]);
[f verts] = reducepatch(isosurface(x,y,z,spd, 30), .2);
h=coneplot(x,y,z,u,v,w,verts(:,1),verts(:,2),verts(:,3),2);
set(h, 'FaceColor', 'cyan', 'EdgeColor', 'none');
[sx sy sz] = meshgrid(80, 20:10:50, 0:5:15);
h2=streamline(x,y,z,u,v,w,sx,sy,sz);
set(h2, 'Color', [.4 1 .4]);
colormap(jet)
box on
axis tight
camproj perspective;
camva(34);
campos([165 -20 65]);
camtarget([100 40 -5])
camlight left;
lighting gouraud
```

step 2 将程序代码保存为"volize.m"，返回MATLAB的命令窗口，输入"volize"，按"Enter"键，得到的图形如图10.28所示。



图10.28　程序运行的结果

在上面的代码中，首先加载MATLAB自带的数据文件wind，该数据文件的默认保存路径为...\MATLAB7.0\toolbox\matlab\general。在程序代码中，首先使用isosurface命令从wind文件选取数据，然后使用patch创建"块"对象p。在后续的程序代码中，使用set函数设置图形对象的各种属性。在程序的最后部分，使用各种照相机命令来设置图形的相机属性。



在上面的第一行程序代码中，首先使用colordef命令设置了图形的颜色。关于该命令的主要信息，可以查看相应的帮助信息。

10.6 综合实例

前面分别介绍了句柄图形的各个部分内容，如果需要绘制个性化和精美的图形，通常需要设置各种图形对象的属性。因此，在本章的最后一节中，将介绍如何综合使用这些相关的命令来绘制复杂的图形。

10.6.1 穿越（fly-through）图形

穿越图形是一种特殊的图形效果，主要通过在三维空间中不断连续变换图形对象的照相机（camera）属性来实现。最后得到的效果就像用户自身在三维空间穿越一样，就像在太空飞船中穿越。之所以被称为穿越，是因为在该图形中用户可以查看到普通视角中无法查看到的内部区域。

为了达到上面所描述的"特殊"图形效果，需要编写程序代码，将图形的相机（camera）属性沿着某个特殊的路径移动，例如将图形的相机（camera）属性沿着x坐标轴移动。而且，为了达到穿越效果，需要同时修改图形对象的位置（CameraPosition）和目标（CameraTarget）属性。

在本综合实例中，通过加载MATLAB自带的wind数据文件，绘制出由风速矢量场所定义的三维图形空间的内部等势表面。用户可以穿越整个三维空间，查看其内部表面区域。由系统自带的wind数据文件描述北美空间的空气势能。

由于该实例比较复杂，下面需要分步骤详细介绍整个动态图形的创建过程。

例 10.15　在 MATLAB 中，绘制 wind 数据文件的穿越图形。

step 1 绘制等势表面。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>>load wind
>>wind_speed = sqrt(u.^2 + v.^2 + w.^2);
>>hpatch = patch(isosurface(x,y,z,wind_speed,35));
>>isonormals(x,y,z,wind_speed,hpatch)
>>set(hpatch,'FaceColor','red','EdgeColor','none');
```

step 2 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图 10.29 所示。

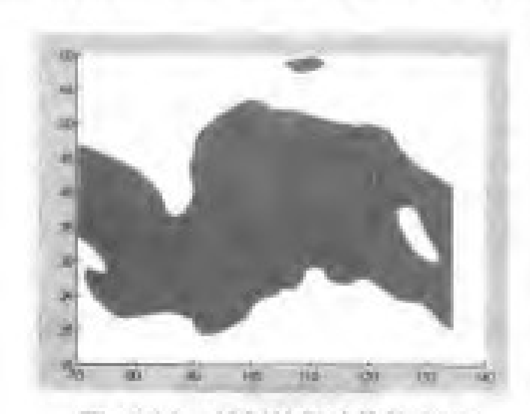


图 10.29　绘制的风速等势表面

step 3 绘制穿越等势表面的圆锥流体。继续在 MATLAB 的命令窗口中输入代码：

```
>> [f vt] = reducepatch(isosurface(x,y,z,wind_speed,45),0.05);
>>daspect([1,1,1]);
>>hcone = coneplot(x,y,z,u,v,w,vt(:,1),vt(:,2),vt(:,3),2);
>>set(hcone,'FaceColor','blue','EdgeColor','none');
```

step 4 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图 10.30 所示。

step 5 定义视角参数。前面步骤中已经创建基础的平面图形，在创建三维图形之前，需要首先定义视角参数。在 MATLAB 的命令窗口中输入下面的代码：

```
>> camproj perspective
```

```
>>camva(25).
```

step 6 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图10.31所示。

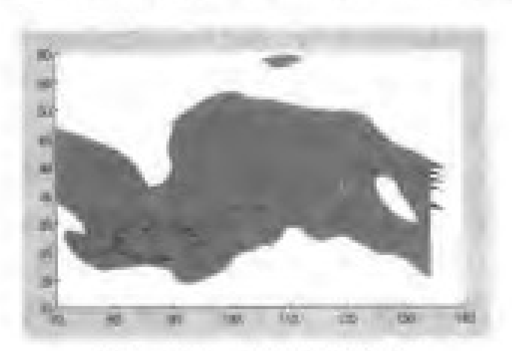


图10.30　绘制圆锥流体

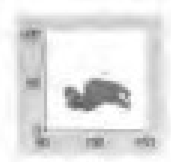


图10.31　设置视角参数



在上面的程序代码中，首先设置了该图形的投影方式是透视，然后设置视角为25°。之所以设置的角度为25°，是为了在后面的步骤中方便查看空间的内部表面。

step 7 指定图形光源。在MATLAB的命令窗口中输入下面的代码：

```
>> hlight = camlight('headlight');
>>set(hpatch,'AmbientStrength',.1,...
      'SpecularStrength',1,...
      'DiffuseStrength',1);
>>set(hcone,'SpecularStrength',1);
>>set(gcf,'Color','k')
```

step 8 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图10.32所示。



图10.32　设置图形的光源属性

在上面的程序代码中，首先在照相机所在的位置创建一个光源，将前面步骤创建的"块"对象中的环境光的光度（AmbientStrength）设置为0.1，由于该数值十分接近0，表明环境光近乎为黑色，然后将"块"对象的镜面反射强度（SpecularStrength）和漫反射强度（DiffuseStrength）都设置为1，因此，"块"对象的反射强度很大。接着，将锥形流体的镜面反射强度（SpecularStrength）也设置为1。在程序的最后，将图形窗的颜色设置为黑色。

step 9 选择图形光源的着色（Render）属性。在MATLAB的命令窗口中输入下面的代码：

```
>> lighting phong
>>set(gcf,'Renderer','zbuffer')
```



之所以需要在这个步骤中设置光源的着色属性，是因为前面步骤中在图形对象中设置了光源。MATLAB提供zbuffer和OpenGL两种着色属性的设置。在本实例中，选择的是zbuffer。关于这些着色属性的详细内容，感兴趣的读者可以查阅MATLAB中相应的帮助文件。

step 10 定义照明路线。在MATLAB的命令窗口中输入下面的代码：

```
>> hsline = streamline(x,y,z,u,v,w,80,30,11);
>> xd = get(hsline,'XData');
>> yd = get(hsline,'YData');
>> zd = get(hsline,'ZData');
>> delete(hsline)
```

在上面的程序代码中，首先创建一个从点（80，30，11）的流线型照明路线，然后通过get函数获取该流线型直线对象的x轴、y轴和z轴的数据，分别保存在数组xd、yd和zd中。最后，将该流线型直线对象删除。

step 11 演示穿越效果。在MATLAB的命令窗口中输入下面的代码：

```
>>for i=1:length(xd)-50
   campos([ xd(i),yd(i),zd(i)])
   camtarget([ xd(i+5)+min(xd)/100,yd(i),zd(i)])
   camlight(hlight,'headlight')
   drawnow
end
```

step 12 查看图形结果。当输入上面的代码后，按"Enter"键，就可以实现最后的穿越效果。由于是动态效果，很难在纸质条件下显示出来，下面分别选取动态效果中的三幅图形，使用户感受到动态效果，图形依次如图10.33至图10.35所示。



图 10.33 动态图形一

在上面的程序代码中，为了演示穿越效果，沿着相同的路径移动照明的位置（position）和目标（target）属性。在本实例中，照明的目标属性比x轴远5个元素值；之所以设置这个比较小的元素值，是为了避免在后面的步骤中照明的位置和目标属性重合。



图 10.34 动态图形二



图 10.35 动态图形三

具体来讲，在程序代码中沿着前面步骤中设置的流线型照明路线，同时更新数据点的照明位置和目标属性，然后沿着该照明路线移动光源，在最后，调用 drawnow 命令来绘制移动后的图形。

本实例属于MATLAB实时动画，这种图形保持图形窗中绝大部分的像素颜色不变，而只更新部分像素的颜色来构成运动图像。



在 MATLAB 中，drawnow 命令的功能是进行屏幕刷新。为了显示实时动画，当新对象属性设置完成后，应该刷新屏幕，使新对象显示出来。drawnow 命令使得 MATLAB 暂停目前的任务序列而去刷新屏幕；如果没有 drawnow 命令，MATLAB 需要等所有的任务序列执行完成后才会去刷新屏幕。

10.6.2 动态反射图形

上面已经介绍设置图形对象的照相机属性来实现特殊的穿越效果。在 MATLAB 中，还可以通过其他方法来创建动态图形。在本小节中，将使用另外的方法来创建动态图形。

在 MATLAB 中，常见的动画有影片动画和实时动画两种。其中影片动画的基本原理如下：首先预先创建图形，将图形存放在缓冲区，然后按照设置的条件逐帧播放。这种动画比较适合于播放比较复杂的画面，但是计算方法的计算量大，占用内存多，而且播放时间短。

在本小节中，将介绍的实例属于另外一种动态图形——实时动画。这种动画比较适合于每次变化较少、图形精度不高的情况。在介绍动态图形创建过程之前，首先介绍一个比较重要的属性——EraseMode。

在 MATLAB 中，实现实时动画的基本原理是，显示新对象、擦除旧对象，同时不破坏背景图片。上面的效果使用其他编程语言比较难实现，但是在 MATLAB 中可以使用系统的 EraseMode 属性来十分便捷地实现该效果。MATLAB 提供了下面四种常见的 EraseMode 属性，其对应的含义如下。

- {normal}：计算整个图形画面的数据，重新绘制整个图形。相对于其他的属性，选用该属性绘制的图形最准确、也最慢。
- none：对图形中所有的对象不作任何的擦除。该绘制模式下绘制的图形不能被正确打印。
- xor：异或方式，对象的绘制和擦除应该由对象颜色和屏幕颜色来决定，只画与屏幕颜色不同的对象数据点，只擦除与屏幕颜色不一致的数据点。
- background：将旧对象的颜色变成背景颜色，实现擦除。

可以根据需要选用合适的擦除模式，根据不同的擦除模式选择合适的着色模式，这样就可以创建自己所需的动画效果。

在详细说明该实例之前，简要介绍创建动画效果的大致步骤：

step 1 绘制初始图形；

step 2 计算出活动对象的新位置，并在新位置中显示该对象；

step 3 擦除原位置中原有对象，刷新屏幕；

step 4 重复上面的两个步骤。

由于该实例比较复杂，下面需要分步骤详细介绍整个动态图形的创建过程。

例 10.16 在 MATLAB 中，绘制动态光线的反射图形。

为了让读者事先了解反射图形的动态性，在介绍创建步骤之前，首先将动态图形的初始图形和过程图形显示给读者。其中，该动态的初始图形如图 10.36 所示。

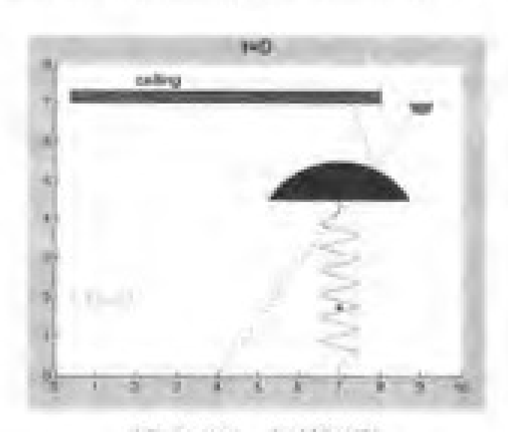


图 10.36 初始图形

该图形将会持续 20 秒的时间，其中当 t = 7 的时候，图形如图 10.37 所示。

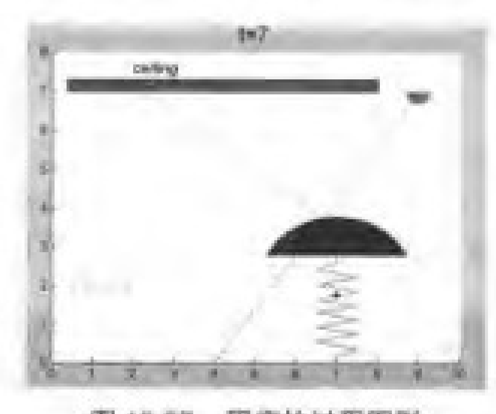


图 10.37 程序的过程图形

在程序的最后还将需要显示该轨迹点的变化图形，如图 10.38 所示。

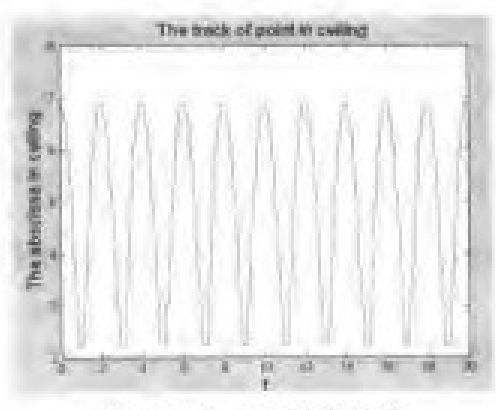


图 10.38　反射点的轨迹

下面将分步骤详细介绍上面的动态效果的创建过程。

step 1 单击 MATLAB 命令窗口工具栏中的 □ 按钮，打开 M 文件编辑器。在 M 文件编辑器中输入下面的程序代码：

```
%---------------- 设置动画图形的初始环境变量 -----------------------------%
clear;clc;close all;
figure;
set(gcf,'DoubleBuffer','on');    % 设置图形的刷新模式
axis([0,10,0,8]);                % 设置图形的坐标轴范围
hold on;
%---------------- 创建图形的顶板(Ceiling)对象 ----------------------------%
fill([0.4,8,8,0.4],[7,7,7.3,7.3],[0.8,0.1,0.7]);    % 创建图形中的横栏
text(2,7.6,'{\it\bf{ceiling}}','fontsize',12);       % 添加横栏的文字说明
```

step 2 查看图形结果。当输入上面的代码后，选择 M 文件编辑器中的 "Debug" ⇨ "Run" 命令，按快捷键 "F5"，查看该部分代码的运行结果，如图 10.39 所示。

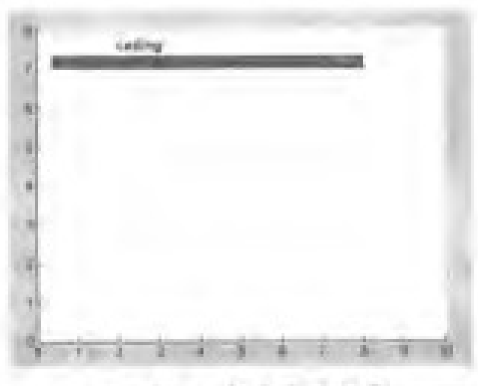


图 10.39　部分程序结果

在本实例中，顶板 Ceiling 是用来接收反射光的工具，因此其长度范围必须满足所有反射光的射程。在上面的程序代码中，使用 fill 命令来创建该顶板对象。在 MATLAB 中，fill(和 fill3) 命令的功能是绘制任意多边形。用户必须保证绘图数据首尾重合，使勾画的多边形封闭。



之所以在上面的步骤中显示部分程序代码的结果，是因为该程序代码比较复杂，为了让读者了解各部分程序的功能，笔者将一边编写代码一边调试代码，得到部分结果。

step 3 返回到 M 文件编辑器中，在上面的代码后面输入程序代码：

```
%-----------------创建图形的光源对象----------------------------------------%
Zz=9+7i+exp(i*linspace(pi,pi*2,20))*0.3;              %定义半圆光源的数字方程
Fc=fill(real(Zz),imag(Zz),'r');                        %绘制半圆光源
set(Fc,'EdgeColor','r');                               %设置光源的边缘颜色
```

step 4 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图 10.40 所示。



图 10.40　添加半圆光源



在上面的程序代码中，首先定义了数组 Zz，该数组中的数值表示的是光源半圆的数据点坐标值，使用的是复数极坐标方法，然后在程序代码中使用 fill 绘制填充半圆图形，最后使用 set 函数设置半圆图形的边缘颜色为红色，得到一个红色的半圆光源。

step 5 返回到 M 文件编辑器中，然后在上面的代码后面输入下面的程序代码：

```
%-----------------创建反射半镜对象----------------------------------------%
Ar=pi/6;
Zx=7+3.5*i+2*exp(i*linspace(Ar,pi-Ar,20));            %设置反射半圆的数学方程
Fg=fill(real(Zx),imag(Zx),'b');
set(Fg,'EdgeColor','b');
```

step 6 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图 10.41 所示。

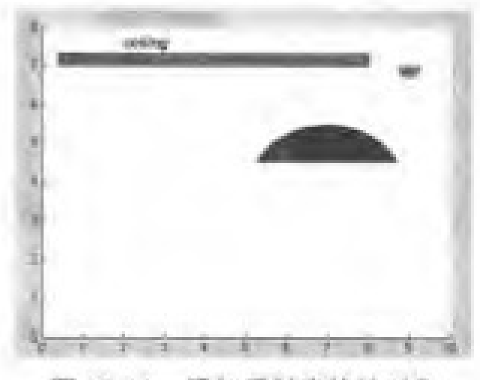


图 10.41　添加反射半镜的对象

上面这段程序代码原理和前段程序代码类似，具体的坐标数值和颜色属性不同。

step 7 返回 M 文件编辑器，在上面的代码后面输入程序代码：

```
%-----------------创建弹簧的球心对象------------------------------------------%
Zr=(3.5-2*cos(Ar))*i+7;
hc=plot(Zr,'k*');   % 球心
p1=9+7i;
p3=4;
```

step 8 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图10.42所示。

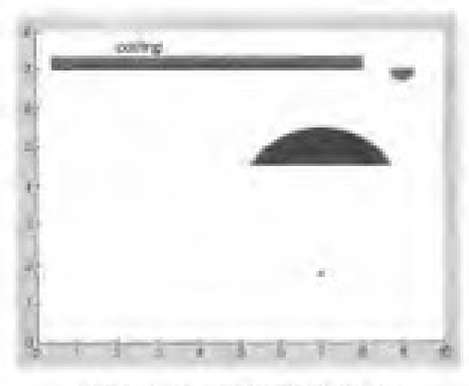


图 10.42　添加球心对象

在上面的程序中，变量p1表示的是光源发射点，其坐标数值为（9,7），p2变量是发射光线的延长线和横坐标的交点坐标。

step 9 返回M文件编辑器，在上面的代码后面输入程序代码：

```
%---------------定义发射光线的数学方程-----------------------------------%
Dtan=abs(angle(p1-Zx)-angle(Zx-p3));              %计算相角差
[a,Kk]=min(Dtan);                                 %计算相角差的最小值
p2=Zx(Kk);                                        %选择放射半圆中的数据点
Hp1=plot([p1,p2],'r');
Hp2=plot([p3,p2],'k:');
```

step 10 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图10.43所示。

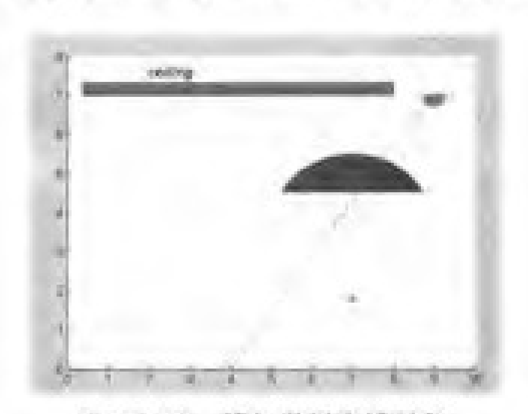


图 10.43　添加发射光线对象

在上面的程序代码中，首先将反射半圆数据数组（Zx）中的数据和光源发射点p1的坐标相减，得到一个新的复数，然后使用angle函数求解该复数的相角；计算反射半圆数据数组（Zx）中的数据和坐标轴数据点p3的坐标相减，得到一个新的复数，然后使用angle函数求解该复数的相角；最后，计算两个相角的差值。

上面步骤计算得到的差值是一个数组，然后使用min函数求解该相角差的最小值和数组的序号kk。接着，选择反射半圆数据数组（Zx）中对应序号kk中的数值p2。在上面程序的最后，绘制发射光线和发射光线的延长线。

step 11 返回M文件编辑器，在上面的代码后面输入程序代码：

```
%--------------定义反射半圆的反射光线的数学方程------------------------%
Af=2*angle(p2-Zr)-angle(p1-p2);
L=(7-imag(p2))/cos(Af-pi/2);
pf=p2+L*exp(i*Af);
Hp3=plot([p2,pf],'r');
```

step 12 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图10.44所示。

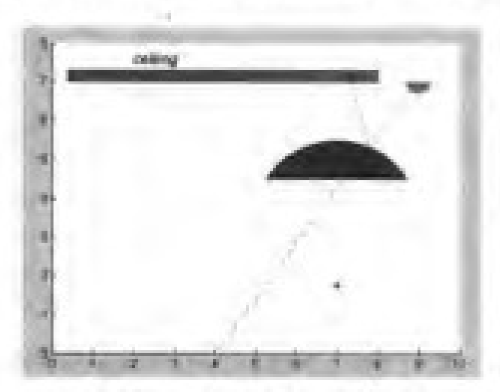


图 10.44　绘制反射光线对象

在上面的程序代码中，首先计算相角Af和复数长度L，然后以变量p2为基础来计算反射光线在光板中的数据的坐标数值，最后使用plot绘制反射光线。

step 13 返回M文件编辑器，在上面的代码后面输入程序代码：

```
%--------------定义弹簧的数学方程以及坐标数值----------------------------%
N=16;
xt=7*ones(1,N);
yt=linspace(0,4.5,N);
Kt=mod(1:N-4,2)-0.5;
xt=xt+[0,0,Kt,0,0];
hpt=plot(xt,yt);
```

step 14 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图10.45所示。

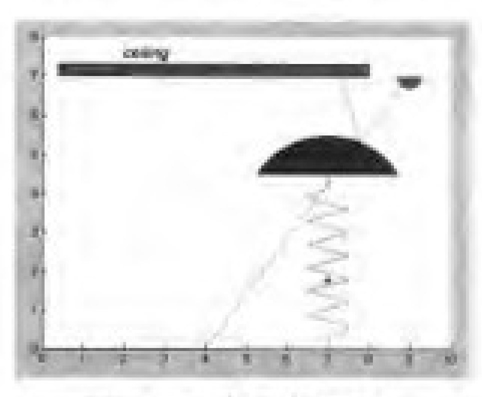


图 10.45　添加弹簧对象

在上面的程序代码中，xt代表的是弹簧对象的横坐标数组，yt代表的是弹簧对象的纵坐标数组，为了实现弹簧的折线效果，引入中间变量Kt。其具体的算法请读者自己演示和分析，mod函数的用法请查阅数值计算的章节。

step 15 返回M文件编辑器，在上面的代码后面输入程序代码：

```
%--------------- 定义图形的初始标题和时间标量 ------------------------------%
hw=title('t=0','fontsize',16);
text(0.4,2,'t\in[0,20]','fontsize',14,'color',[0.2,0.8,0.6]);
```

step 16 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图10.46所示。

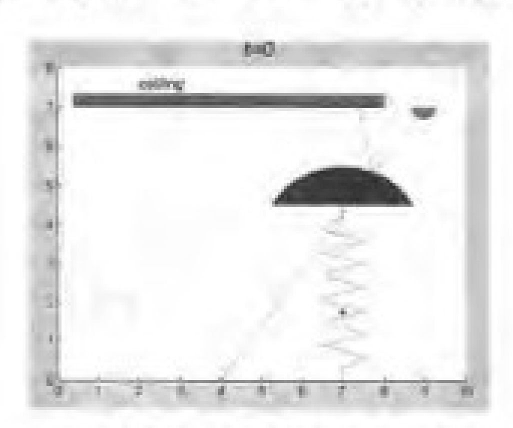


图 10.46　添加图形的标题和说明

在上面的程序代码中，首先使用title命令创图形的标题对象，然后使用text命令创建图形的时间说明文字。其中时间说明文字中使用了特殊数学符号，在MATLAB中，该数学符号属于图形标识中的特殊字符。该具体字符是MATLAB从Tex字符集中摘引了包括希腊字母在内的100多个特殊字符。关于这些特殊字符，可以查看MATLAB的相关帮助文件。



到本步骤为止，已经创建该动态图形的初始图形界面，也就是当系统时间t=0时刻的图形界面。各种图形对象已经绘制完成，同时显示了相应的说明文字。从本步骤开始，将需要创建其他时刻的动态图形界面。

step 17 返回M文件编辑器，在上面的代码后面输入程序代码：

```
%--------------- 定义图形动画的初始参量 ------------------------------%
t=0;              % 控制时间
pg=real(pf);      % 记录反射点的轨迹
dt=0.04;          % 时间的单位递加量
pause(0.1);       %  做一停留
%--------------- 定义图形动画的循环程序代码 ------------------------------%
while t<20;
    t=t+dt;
    pd=7+4*i+cos(pi*t)*i*0.5;
    Zx=pd-2*cos(Ar)*i+2*exp(i*linspace(Ar,pi-Ar,60));
    set(Fg,'XData',real(Zx),'YData',imag(Zx));
    Zr=pd-2*cos(Ar)*i;
    set(hc,'XData',real(Zr),'YData',imag(Zr));
    Dtan=abs(angle(p1-Zx)-angle(Zx-p3));
    [a,Kk]=min(Dtan);p2=Zx(Kk);
```

```
    set(Hp1,'XData',real([p1,p2]),'YData',imag([p1,p2]));
    set(Hp2,'XData',real([p3,p2]),'YData',imag([p3,p2]));
    Af=2*angle(p2-Zr)-angle(p1-p2);
    L=(7-imag(p2))/cos(Af-pi/2);
    pf=p2+L*exp(i*Af);
    pg=[pg,real(pf)];
    set(Hp3,'XData',real([pf,p2]),'YData',imag([pf,p2]));
    yt=linspace(0,min(imag(Zx)),N);
    Kt=mod(1:N-4,2)-0.5;
    set(hpt,'YData',yt);
    set(hw,'String',['t=',num2str(t)]);
    pause(0.05);
end
```

step 18 查看图形结果。输入程序代码后，按“Enter”键，得到的图形如图 10.47 和图 10.48 所示。

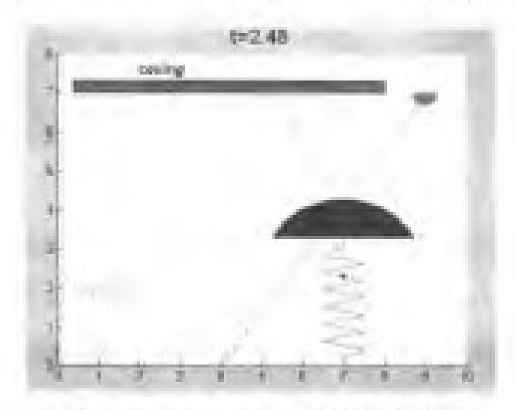


图 10.47　t = 2.48 时的图形界面

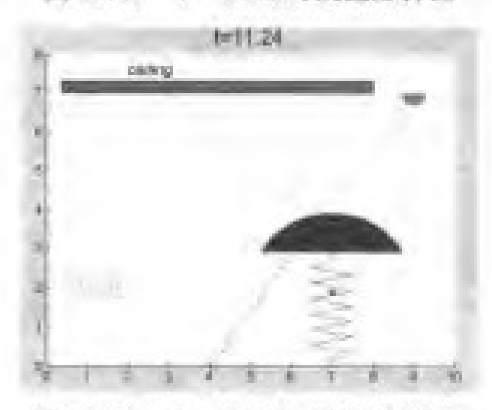


图 10.48　t = 11.24 时的图形界面

从上面的程序结果中可以看出，当 MATLAB 开始运行循环程序时，图形标题将会显示对应的运行时刻，而弹簧对象将开始上下振动，发射光将会透射到反射透镜表面不同的位置，因此将会产生不同的反射光线。程序将一直循环运行下去，直到时间变量 t 达到用户指定的 20 秒，最后的结果在前面的章节中已经给出，这里就不重复了。



如果读者细心阅读上面while循环体内的程序代码，就会发现该代码和前面步骤中创建初始图形界面的代码十分相似，只是添加了控制变量t。因此，在这里就不介绍其代码含义了，请读者对比前面步骤中的代码自行分析。

step 19 返回 M 文件编辑器，在上面的代码后面输入程序代码：

```
%--------------------创建一个新的图形窗口----------------------------------%
figure;
plot(linspace(0,20,length(pg)),pg);          %绘制光线反射点的坐标轨迹
title('The track of point in ceiling','fontsize',14);
xlabel('{\itt}','fontsize',14);
ylabel('The abscissa in ceiling','fontsize',14)
```

step 20 查看图形结果。输入程序代码后，按"Enter"键，得到的图形如图 10.49 所示。

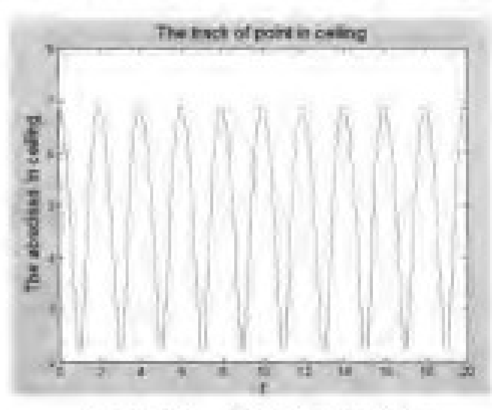


图 10.49　反射点轨迹图形

step 21 保存上面步骤的代码。前面的步骤已经完成动态图形的创建工作，可以将其保存为 M 文件。在本例中将其保存为"reflash.m"文件，保存后的结果如图 10.50 所示。

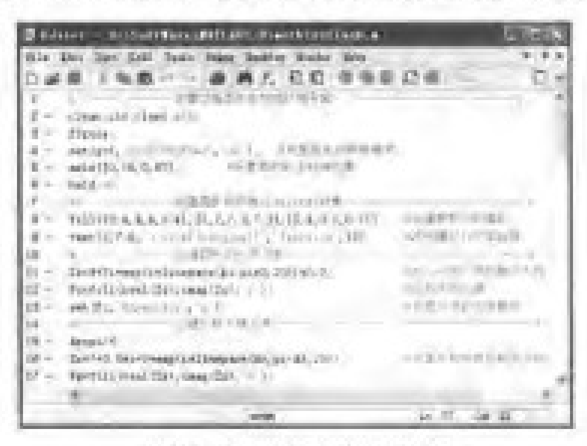


图 10.50　保存代码程序



当保存创建动态图形的代码后，在后面的步骤中运行该动态图形时，就可以直接在命令窗口中输入"reflash"，直接查看动态图形。

10.7 小结

在本章中，向读者介绍了句柄图形的内容，主要包括句柄图形体系、图形句柄的操作、图形对象的操作、高层绘图命令和坐标轴对象等，这些内容可以认为是"数据和函数的可视化"的后续内容。把第9和第10两章内容结合起来可以对MATLAB中的图形设计有一个全面的认识。在后面的章节中，将介绍图形用户界面的内容。

◆　　◆　　◆

第11章　图形用户界面（GUI）制作

本章包括

- ✦ 使用 GUIDE 创建 GUI
- ✦ 使用 M 文件创建 GUI
- ✦ 创建自定义菜单
- ✦ 创建现场菜单
- ✦ 添加 GUI 的图形控件

有过编程经历的读者也许都会有类似的经验，一个可以发布的应用程序通常都需要有一个友好的图形用户界面（Graphical User Interface）。程序的用户界面是用户与计算机程序的交互方式，用户通过键盘、鼠标等输入设备与计算机交换信息。图形用户界面（简称GUI）是包含图形对象，如窗口、图标、菜单和文本的用户界面。用户以某种方式选择或激活这些对象，会引起动作或发生变化，例如调用计算程序或者绘图等。

对于第一次接触GUI概念的读者，前面的文字介绍也许比较陌生。其实大多数读者已经接触过一些MATLAB定制的GUI对象，例如，前面章节介绍过的演示程序demo就是典型的图形用户界面。可以在命令窗口中输入demo来调用相应的演示程序，然后在图形界面中使用鼠标进行选择和点击，查看相应的演示程序结果。

在本章中，首先用两个简单的例子来介绍如何使用M文件和GUIDE工具来创建GUI，介绍创建GUI的基本原则和步骤；然后，分节介绍使用M函数文件创建GUI的菜单对象和控件对象的方法，加强对图形界面的工作原理的理解；最后，将分步骤介绍创建比较复杂GUI对象的过程，创建这个对象将需要综合应用前面章节创建图形的各种知识。因此，本章内容可以帮助读者复习前面章节的内容。

11.1　图形用户界面概述

在通常情况下，开发实际的应用程序时应该尽量保证程序的界面友好，因为程序界面是应用程序和用户进行交互的环境。在当前情况下，使用图形用户界面是最常用的方法。提供图形用户界面可使用户更方便地使用应用程序，不需要了解应用程序怎样执行各种命令的，只需要了解图形界面组件的使用方法；同时，不需要了解各种命令是如何执行的，只要通过用户界面进行交互操作就可以正确执行程序。

在MATLAB7.0中，图形用户界面通常是一种包含多种图形对象的界面，典型的图形界面包括图形显示区域，功能按钮控件以及用户自定义的功能菜单等。为了让界面实现各种功能，需要对各个图形对象进行布局和事件编程。这样，当用户激活对应的GUI对象时，就能执行相应的事件行为。最后，必须保存和发布自己创建的GUI，使得用户可以应用GUI对象。



在创建图形用户界面的时候，需要遵循不同的设计要求，因此创建出的图形界面也会千差万别。这并不意味着创建图形界面没有统一的标准。在设计图形界面时，除了保证程序功能之外，还需要遵循一些统一的标准，这些标准将在后面详细介绍。

在介绍使用MATLAB进行M文件编程时，曾经介绍过，在MATLAB中，所有对象都可以使用M文

件进行编写。GUI也是一种MATLAB对象，因此，可以使用M文件来创建GUI。同时，使用M文件创建GUI的方法也是最基础的，使用其他方法创建GUI图形界面时，实现图形控件的各种功能时，也需要编写相应的程序代码。最后，了解创建GUI对象的M程序代码，也可以帮助用户理解GUI的各种组件和图形对象控件的常用属性等内容。



关于使用M文件创建图形用户界面的方法，将在后面详细介绍调用方法和格式，这里就不详细展开介绍了。

除了使用M文件来创建GUI对象之外，MATLAB还为用户开发图形界面提供一个方便高效的集成开发环境：MATLAB图形用户界面开发环境（MATLAB Graphical User Interface Development Environment），简称为GUIDE。其主要是一个界面设计工具集，MATLAB7.0将所有GUI所支持的用户控件都集成起来，同时提供界面外观、属性和行为响应方法的设置方法。除了可以使用GUIDE创建GUI之外，还可以将设计好的GUI界面保存为一个FIG资源文件，同时自动生成对应的M文件。该M文件包含了GUI初始化代码和组建界面布局的控制代码。

使用GUIDE创建GUI对象执行效率高，可以交互式地进行组件布局，还能生成保存和发布GUI的对应文件。

- **FIG文件**：该文件包含GUI图形窗口及其子对象的完全描述，包含所有相关对象的属性信息，可以调用hgsave命令或者使用M文件编辑器的"File"→"Save"命令生成该文件。FIG文件是一个二进制文件，包含系列化的图形窗口对象。所有对象的属性都是用户创建图形窗口时保存的属性。该文件最主要的功能是对象句柄的保存。
- **M文件**：该文件包含GUI设计、控制函数及控件的回调函数，主要用来控制GUI展开时的各种特征。该文件基本上可以分为GUI初始化和回调函数两个部分，控件的回调函数根据用户与GUI的具体交互行为分别调用。应用程序M文件使用openfig命令来显示GUI对象，但是该文件不包含用户界面设计的代码，对应代码由FIG文件保存。



关于使用GUIDE创建GUI对象的具体步骤和对应的注意事项，将在后面利用具体的实例详细介绍，这里就不展开介绍了。

11.2 使用M文件创建GUI对象

在介绍GUI对象的详细内容之前，为了让读者了解创建GUI对象的两种基本方法的步骤，在本小节中，将使用两个简单的实例来说明各种方法的特点。

本节将介绍如何使用M文件来创建一个简单的GUI对象，该GUI对象中不包含GUI菜单和控件，其与用户之间的互动通过键盘和鼠标操作来实现。对于这种类型的GUI对象，最好使用M文件来直接编写而不适合使用GUIDE来创建。

由于该实例比较复杂，下面分小节详细介绍创建该GUI对象的步骤。

11.2.1 编写程序代码

例11.1 在MATLAB中，创建一个三维齿轮GUI对象。在该GUI界面中，用户可以使用鼠标来

控制齿轮运动的方向，使用方向按键来控制查看该三维对象的角度。

step 1 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的 "File" ⇨ "New" ⇨ "M-file" 命令，打开一个空白的 M 文件编辑器，然后在 M 文件编辑器中输入下面的代码：

```
function gear3d(varargin)
%   GEAR3D   GUI example of 3D gear.
%   GEAR3D will pop up a GUI example of 3D gear. The gear rolls on a
geared
%   ground based on the mouse location. It uses the x-location of the
mouse
%   pointer for the gear location. Use the arrow keys to change the
view.
%   Press SPACEBAR to reset the view.
%
%
%   调用命令示例
%     GEAR3D('teeth', 30)    - 默认值是 50
%     GEAR3D('spokes', 4)    - 默认值是 8
%     GEAR3D('ratio', 2)     - 默认值是 3
%   该命令只接受正值的参数

% ------------------------参数的默认数值------------------------------%
% Number of teeth
numteeth  = 50;
% Number of spokes
numspokes = 8;
% Gear ratio
rr        = 3;
% ----------------------处理函数的参数属性----------------------------%
if mod(nargin, 2) == 1
  error('Optional arguments must come in pairs.');
end
if nargin
  opt = varargin(1:2:end);
  val = varargin(2:2:end);
  validOpts = {'teeth', 'spokes', 'ratio'};
  for iArg = 1:length(opt)
    id = strmatch(lower(opt{iArg}), validOpts);
    if isempty(id)
       error('Invalid option. Valid options: ''teeth'', ''spokes'',
''ratio''');
    else
      switch strmatch(lower(opt{iArg}), validOpts)

        case 1
          if isnumeric(val{iArg}) & length(val{iArg}) == 1
            numteeth = round(abs(val{iArg}));
          end
                case 2
          if isnumeric(val{iArg}) & length(val{iArg}) == 1
            numspokes = round(abs(val{iArg}));
          end
                case 3
          if isnumeric(val{iArg}) & length(val{iArg}) == 1
```

```
            rr = round(abs(val( iArg) ));
          end
             end
      end
    end
end
%-----------------------------计算参数的数值-------------------------%
% 半径数值
r1 = 1;
r2 = 3;
r3 = 10;
r4 = 11;
r5 = 13;
r6 = 12;
% 中心齿轮的半径数组合 i
r = (r5 + r6) / 2;
% Height of gear teeth
h = r5 - r6;

l = r * rr;
l1 = l  - h/2;
l2 = l  + h/2;
l3 = l2 + 2;
%----------------------------定义齿轮对象的数据-----------------------%
[ x0,y0,z0]        = cylinder([ r6] , numteeth*4);  % 齿轮的凹槽
[ x1,y1,z1]        = cylinder([ r1, r1, r2, r2, r1] , numteeth*4);  % 轮辐
[ x2,y2,z2]         = cylinder([ r3, r3, r4, r4, r5, r5, r4, r4, r3] ,
numteeth*4);  % 轮齿
z1([ 1, 4, 5] , :) =  2;
z1(2:3, :)         = -2;
z2([ 1, 8, 9] , :) =  2;
z2(2:3, :)         = -2;
z2(4:5, :)         = -1;
z2(6:7, :)         =  1;
x2(5:6,1:4:end)  = x0(1:2,1:4:end);
x2(5:6,2:4:end)  = x0(1:2,2:4:end);
y2(5:6,1:4:end)  = y0(1:2,1:4:end);
y2(5:6,2:4:end)  = y0(1:2,2:4:end);
%----------------------------定义轮辐对象的数据-----------------------%
interval  = round(length(x1) / numspokes);
for id = 1:4
  x1(3:4, id:interval:end) = x2(1:2, id:interval:end);
  y1(3:4, id:interval:end) = y2(1:2, id:interval:end);
end
%----------------------------定义底面对象的数据-----------------------%
[ x3, y3, z3]         = cylinder([ l2, l2, l3, l3, l2] , numteeth * rr * 4);
[ x4, y4, z4]          = cylinder(l1, numteeth * rr * 4);
z3([ 1 4 5] , :)        =  4;
z3([ 2 3] , :)          = -4;
x3([ 1 2 5] , 1:4:end) = x4([ 1 1 1] , 1:4:end);
x3([ 1 2 5] , 2:4:end) = x4([ 1 1 1] , 2:4:end);
y3([ 1 2 5] , 1:4:end) = y4([ 1 1 1] , 1:4:end);
y3([ 1 2 5] , 2:4:end) = y4([ 1 1 1] , 2:4:end);
%---------------------------将底面设置为半圆--------------------------%
len             = round(length(x3) / 2);
x3(:, len:end) = [];
```

```
y3(:, len:end) = [];
z3(:, len:end) = [];
```

在上面的程序代码中，首先定义了默认参数数值，设置程序代码处理用户调用函数的参数处理工作，然后分别定义了各种图形对象的绘制数据，为后面步骤中创建图形对象打好基础。



上面程序代码中使用了多种编写代码结构和功能，对上面代码有疑问的地方，请查看本书中关于M文件编程的章节内容。为了帮助读者了解程序代码的功能，在对应代码的上方添加了对应的注释文字。

step 2 返回 M 文件编辑器，输入下面的代码：

```
%-----------------------------创建图形对象--------------------------------%
fH = findobj('Type', 'figure', 'Tag', 'solidmodelGUI');
if ishandle(fH)
  close(fH);
end
% 创建图形对象的位置
fH = figure(...
  'Name'         , sprintf('3D Gear: View = [ %3.0f, %3.0f] ', 0, -50), ...
  'NumberTitle', 'off', ...
  'Units'        , 'normalized', ...
  'Position'     , [ .1, .1, .8, .8] , ...
  'Tag'          , 'solidmodelGUI');
% 设置图形对象的坐标轴属性
axes(...
  'Units'    , 'normalized', ...
  'Position', [ 0, .9, 1, .1] , ...
  'XLim'     , [ -1, 1] , ...
  'YLim'     , [ -1, 1] , ...
  'Visible' , 'off');
% 添加文字提示信息
text(0, 0, ...
  { 'Move the mouse left and right within the figure window to move the
gear.',...
     'Use ARROW keys to change the view. SPACE to reset view.'} , ...
  'HorizontalAlignment' , 'center', ...
  'VerticalAlignment'   , 'middle', ...
  'Color'               , 'red');
axH = axes('Units', 'normalized', ...
  'Position', [ 0, 0, 1, .9] );
%-----------------------------绘制齿轮对象-----------------------------%

%-----------------------------绘制齿轮--------------------------------%
gearH(1) = surface(x1,y1,z1, ...
  'EdgeColor', [ .3, .3, .3] , ...
  'EdgeAlpha',  0, ...
  'FaceColor', [ .5, .5, 1] );
%-----------------------------绘制轮齿--------------------------------%
gearH(2) = surface(x2,y2,z2, ...
  'EdgeColor', [ .3, .3, .3] , ...
  'EdgeAlpha', .1, ...
```

```
  'FaceColor', [ .5, .5, 1] );
%------------------------------绘制底面----------------------------------%
surface(x3,y3,z3, ...
  'EdgeColor', [ .3, .3, .3] , ...
 'EdgeAlpha', .1, ...
  'FaceColor', [ .5, 1, .5] , ...
  'FaceAlpha', .5);
%------------------------------设置图形的视角----------------------------%
set(axH, 'View', [ 0, -50]);
axis equal manual off;
set(axH, 'CameraViewAngleMode', 'manual');
%------------------------------添加光源----------------------------------%
light('Position', [ 50 , 0, -20]);
light('Position', [ -50, 0, -20]);
```

step 3 查看程序代码的图形结果。上面的程序代码的功能是绘制GUI对象的初始图形，选择M文件编辑器中的"Debug" ⇨ "Run"命令，检测前面程序的结果，如图11.1所示。

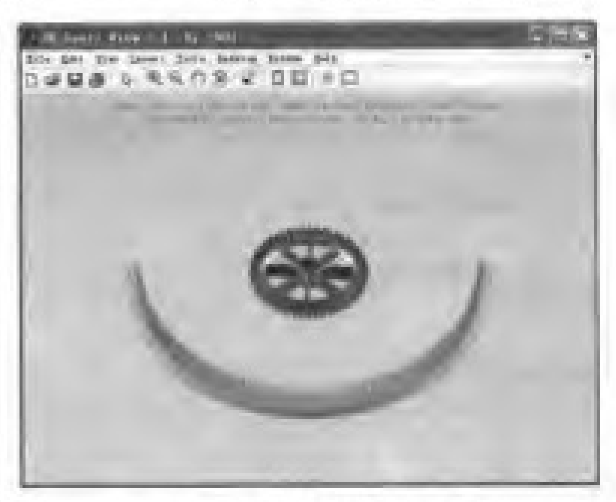


图 11.1 GUI 的初始图形结果



从上面的结果中可以看出，创建GUI的初始图形结果使用的方法和程序代码和前面章节中创建句柄图形一样，可以参考对应章节的内容。

step 4 返回M文件编辑器，输入下面的代码

```
%------------------------------计算图形大小并传递参数--------------------%
sz1 = size(x1);
sz2 = size(x2);
set(fH, ...
  'Visible'                , 'on', ...
  'KeyPressFcn'            , {@myKeyPressFcn, axH} , ...
  'WindowButtonMotionFcn', {@myMotionFcn, gearH, x1, x2, y1, y2, r, l,
sz1, sz2});
%------------------------------设置齿轮的初始位置------------------------%
myMotionFcn(gcf, [], gearH, x1, x2, y1, y2, r, l, sz1, sz2);

%----------------myKeyPressFcn函数 单击方向键调整视角------------------%
function myKeyPressFcn(obj, edata, axH)
```

```
k = get(obj, 'CurrentKey');
aZeL = get(axH, 'View');

switch lower(k)
  case 'uparrow'
    aZeL(2) = min([ aZeL(2) + 10,  90] );
  case 'downarrow'
    % -90 doesn't work well when rotated left or right
    aZeL(2) = max([ aZeL(2) - 10, -89.9999] );
  case 'leftarrow'
    aZeL(1) = max([ aZeL(1) - 10, -90] );
  case 'rightarrow'
    aZeL(1) = min([ aZeL(1) + 10,  90] );
  case 'space'
    aZeL    = [ 0, -50] ;
end
set(axH, 'View', aZeL);
set(obj, 'Name', sprintf('3D Gear: View = [ %3.0f, %3.0f] ', aZeL));
%-------------------myMotionFcn函数：使用鼠标来调整齿轮--------------------%
function myMotionFcn(obj, edata, gearH, x1, x2, y1, y2, r, l, sz1, sz2)
pt = get(obj, 'CurrentPoint');
d = l - r;
x0 = -d + pt(1) * d * 2;
y0 = sqrt(abs(d^2 - x0^2));
th1 = atan2(x0, y0) - pi/2;
th = (th1 - (th1 * (1 / r)));
%----------------------------旋转齿轮对象--------------------------------%

%----------------------------设置旋转角度矩阵-----------------------------%
cosa = cos(-th);
sina = sin(-th);
rot  = [ cosa, -sina; sina, cosa]';
newxy1   = [ x1(:), y1(:)] ;
newxy2   = [ x2(:), y2(:)] ;
newxy1   = newxy1 * rot;
newxy2   = newxy2 * rot;
newx1    = x0 + reshape(newxy1(:, 1), sz1);
newy1    = y0 + reshape(newxy1(:, 2), sz1);
newx2    = x0 + reshape(newxy2(:, 1), sz2);
newy2    = y0 + reshape(newxy2(:, 2), sz2);
%----------------------------设置对象的新位置-----------------------------%
set(gearH, {'XData', 'YData'}, {newx1, newy1; newx2, newy2} );
```



在上面的程序代码中，为了帮助读者了解程序代码的含义，在每段程序代码的上方都添加了对应的注释文字，关于具体的程序代码的含义请读者自行分析。

11.2.2 运行程序代码

延续上面小节的步骤。

step 1 查看程序代码的结果。在上面的程序代码中，设置了控制图形对象的所有代码，选择M文件编辑器中的"Debug" ⇨ "Run"命令，检测前面程序的结果，如图11.2所示。



图 11.2　使用鼠标控制齿轮的运动



从上面的程序代码中可以看出，在初始情况下，齿轮处在整个底面对象的中部，当添加相应的行为程序代码后，使用鼠标就可以控制齿轮的运动情况。用户将齿轮在整个半圆底面中滚动，实现互动的动态图形。

step 2 使用方向键来控制图形。除了可以用鼠标实现互动之外，还可以用方向键控制查看图形的视角，如图 11.3 所示。



从上面的结果可以看出，编写对应的事件函数后，在相应的程序代码中添加回调参数，就可以实现用户和界面的动态交互，这是使用M文件创建GUI对象的核心内容。



图 11.3　使用方向键控制视角

step 3 将上面的代码保存为 "gear3d.m" 文件， 在命令窗口中输入 "GEAR3D('teeth', 60,'spokes', 6, 'ratio', 4)" 命令，然后按 "Enter" 键，得到的图形如图 11.4 所示。

在上面的命令行中，除了调用前面步骤定义的gear3d函数之外，还设置了用户自己设置的图形参数值，然后根据对应的参数绘制图形。



图 11.4　使用自定义参数绘图



根据本例中的各种命令的含义和作用，读者在入门阶段可以不必深究，相应的程序代码在本章的后面内容中将会分别详细介绍。

11.3 使用 GUIDE 创建 GUI 对象

前面已经介绍过，在 MATLAB 中 GUIDE 提供了多种设计模板，用户可以很轻松地定制属于自己的 GUI 对象，同时自动生成对应的 M 文件框架，这样就简化了 GUI 应用程序的创建工作。用户可以直接使用该框架来编写自己的函数代码，因为 GUIDE 模板中包含了相关的回调函数，可以打开对应的 M 文件，查看工作方式或者修改函数，实现用户所需要的功能。

本节将使用一个简单实例来介绍如何使用 GUIDE 创建 GUI 对象。

11.3.1 启动 GUIDE

在本小节中，将使用 GUIDE 来定制 GUI 对象的界面，使用 M 文件来编写对应的事件程序，完成整个 GUI 的创建工作，该 GUI 的最终结果如图 11.5 所示。

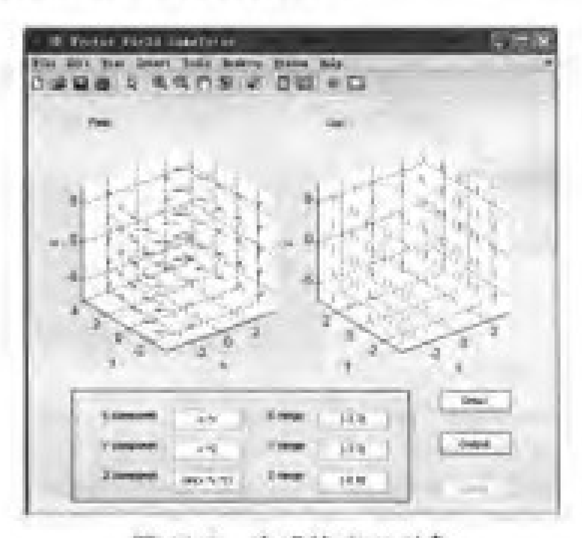


图 11.5　完成的 GUI 对象

在GUI对象中，可以在文本框中输入三维函数的分量，绘制该三维函数的向量场和旋度场图形，同时可以选择输出计算的结果。



尽管上面的GUI对象并不复杂，但是包括了多种GUI的控件，例如按钮、编辑框、组合框、坐标轴和静态文字等，因此使用该实例可以演示GUI的一般创建方法和过程。

下面分步骤来介绍如何创建该GUI对象。

例 11.2 在MATLAB中，创建上面的GUI对象。

step 1 启动GUIDE。选择MATLAB工作界面上的菜单栏中的"GUIDE"快捷菜单，启动MATLAB中的GUIDE，如图11.6所示。

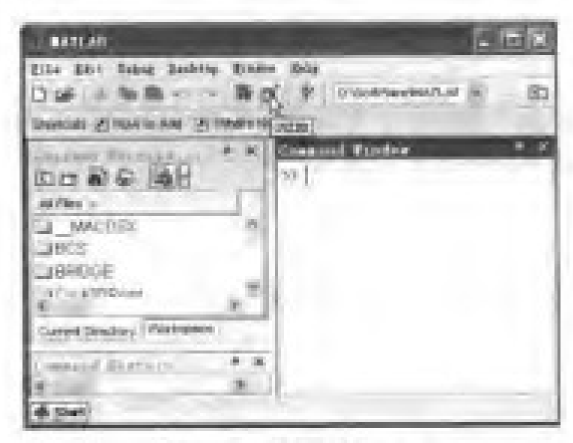


图 11.6 启动 GUIDE

step 2 查看GUIDE。当选择对应的菜单后，就可以打开"GUIDE Quick Start"对话框，如图11.7所示。

在上面的模板设计界面中，可以选择创建新的GUI对象或者打开原来的GUI对象。在创建新的GUI对象中，MATLAB7.0中为用户提供空白模板（Black GUI）、带有控件的模板（GUI with Uicontrols）、带有坐标轴和菜单的模板（GUI with Axes and Menu）和问答式对话框（Modal Questions Dialog）四种模板。其中空白模板是MATLAB中的默认模板，在本实例中将选用该模板创建GUI对象。



图 11.7 GUI 模板设置界面



除了可以使用菜单来打开GUIDE之外，还可以在命令窗口中输入GUIDE命令来启动GUIDE。如果GUIDE已经启动，可以选择菜单栏中的“File”⇨“New”选项，打开新的GUIDE。

step 3 打开空白模板。在“GUIDE Quick Start”对话框中选择“Black GUI(Default)”选项，单击“OK”按钮，MATLAB会显示GUIDE初始化对话框，如图11.8所示。



图 11.8 初始化对话框

当MATLAB完成GUIDE初始化之后，就会显示GUI的空白模板，如图11.9所示。



图 11.9 空白模板的编辑界面

step 4 设置模板的显示属性。选择空白模板菜单栏中的“File”⇨“Preferences”选项，打开“Preferences”对话框，然后在其中选择“GUI”选项，选中“Show names in component palette”选项，显示控件面板中各个控件的名称，如图11.10所示。

step 5 查看修改后的模块。单击“Preferences”对话框的“OK”按钮，就可以得到修改后的模板，如图11.11所示。



图 11.10 设置模板的显示属性



图 11.11 修改后的空白模板



之所以需要显示控件面板中各个控件的名称，是因为对于那些首次接触 MATLAB 的 GUI 设计过程的读者来说，显示控件名称可以帮助他们熟悉相应的控件。

11.3.2 添加控件组件

延续上面小节的步骤。

step 1 向面板中添加"坐标轴"控件。从"控件面板"中选择"Axes"对象，将其拖动到空白模板中合适的位置上，得到的结果如图 11.12 所示。

在 MATLAB 中，坐标轴控件的主要功能是使用户的 GUI 对象可以显示图像对象，用户可以为坐标轴控件设置外观和行为的参数。



在本实例中，不需要为坐标轴控件设置参数，直接使用 MATLAB 中的默认属性就可以很好地显示该实例中的所有图形对象。

step 2 复制"坐标轴"控件。选中上面步骤中添加的"Axes"控件，单击鼠标右键，在弹出的快捷菜单中选择"Duplicate"选项，复制上面步骤添加的"坐标轴"控件，得到的结果如图 11.13 所示。

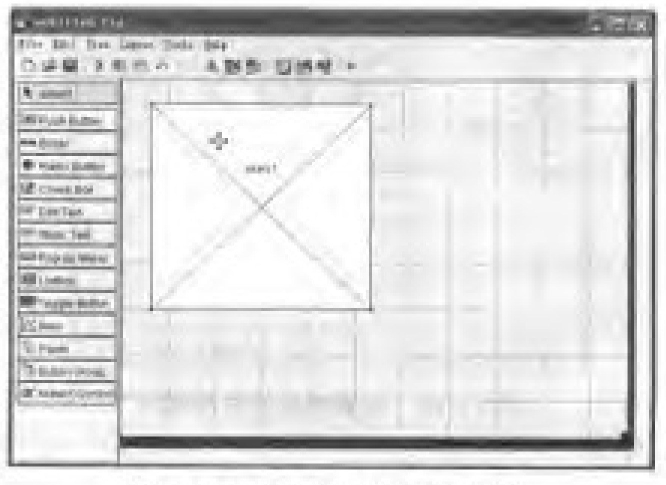


图 11.12 添加"坐标轴"控件

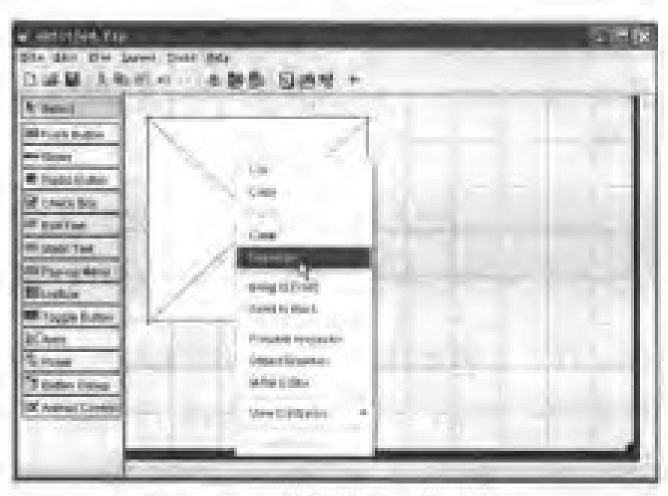


图 11.13 复制"坐标轴"控件

step 3 查看复制后的"坐标轴"控件。当复制"坐标轴"控件后，适当移动复制后控件的位置，得到的结果如图 11.14 所示。

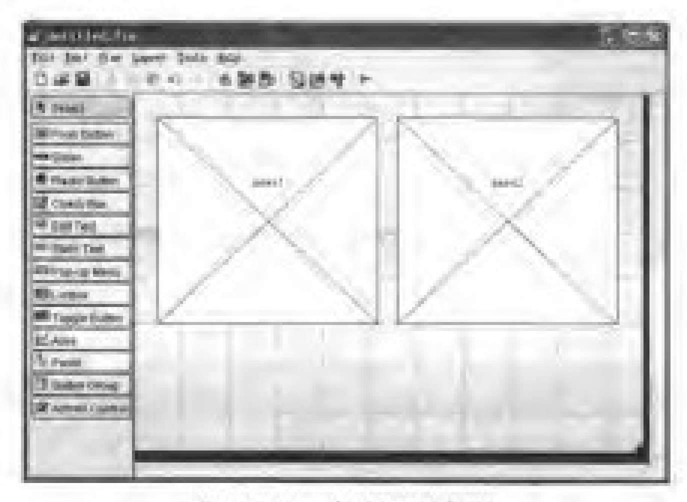


图 11.14 复制后的控件

step 4 添加"组合框（panel）"控件。在本实例中，需要添加绘制图形的各种参数，为了便于管理，需要将这些参数放置在组合框控件中。在控件面板中选择"Panel"控件，然后将其添加到面板中，如图 11.15 所示。

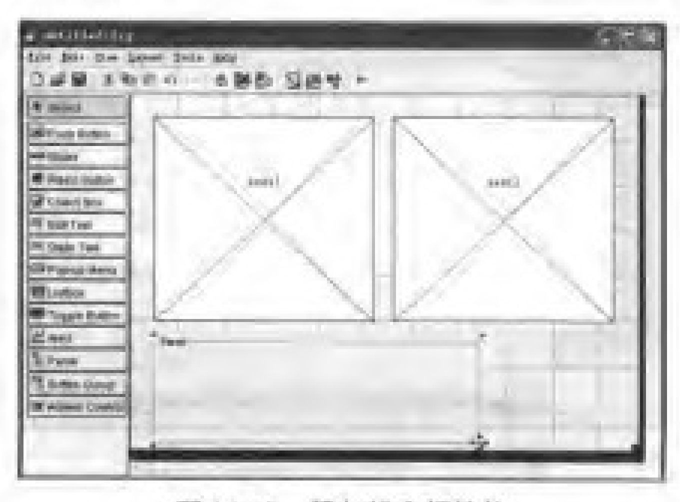


图 11.15 添加组合框控件

在MATLAB中，组合框是图形窗口中的一个封闭区域，它可以将相关联的控件（例如一组单选按钮或者一组编辑框等）组合在一起，使得图形窗口变得更加容易理解。该组合框的主要属性包括标题和边框。



在默认情况下，组合框的边框类型是"etchedin"，也就是内嵌的蚀刻效果；同时其标题为"Panel"，可以通过"属性检查器（Property Inspector）"来修改这些属性。

step 5 添加"编辑框（Edit Text）"控件对象。选择控件面板中的"Edit Text"对象，然后将其添加到上面步骤中添加的"Panel"对象中，如图11.16所示。

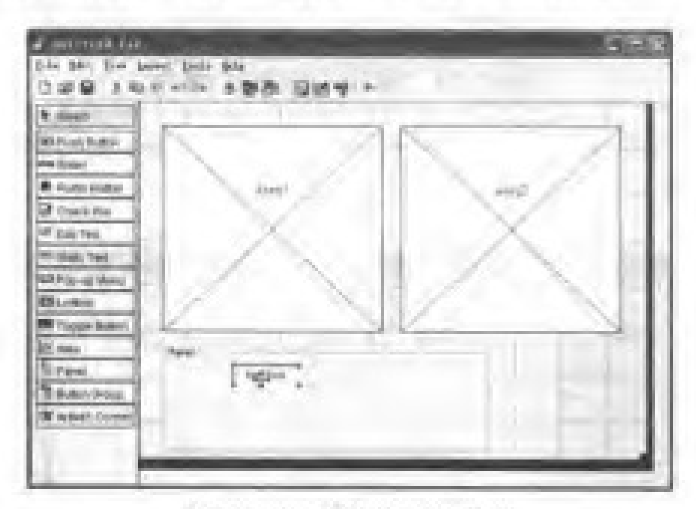


图11.16　添加编辑框控件



在MATLAB中，编辑框控件的功能是控制用户编辑或者修改字符串的文本区域，该对象的String属性包含了用户输入的文本信息。在Windows操作系统中，单击图形窗口不会引发编辑框的回调函数的执行。

step 6 添加"静态文本（Static Text）"控件对象。选择控件面板中的"Static Text"对象，然后将其添加到上面步骤中添加的"Panel"对象中，如图11.17所示。

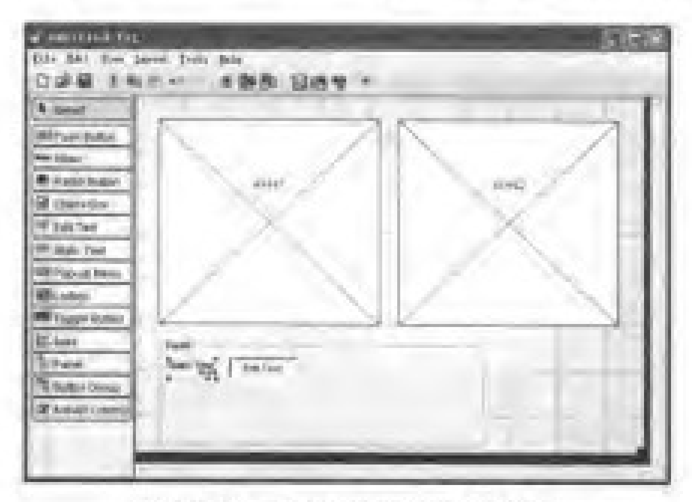


图11.17　添加静态文本控件对象



在MATLAB中，静态文本控件的功能是用做其他控件的标签，该控件对象和编辑框控件的主要区别在于，用户不能在交互的状态下修改其文字或者调用其相应的回调函数。在本实例中，该控件的功能在于显示输入参数的标签。

step 7 复制"编辑框（Edit Text）"和"静态文本（Static Text）"控件对象。使用前面步骤介绍的方法，复制前面步骤添加的"编辑框"和"静态文本"控件，结果如图11.18所示。

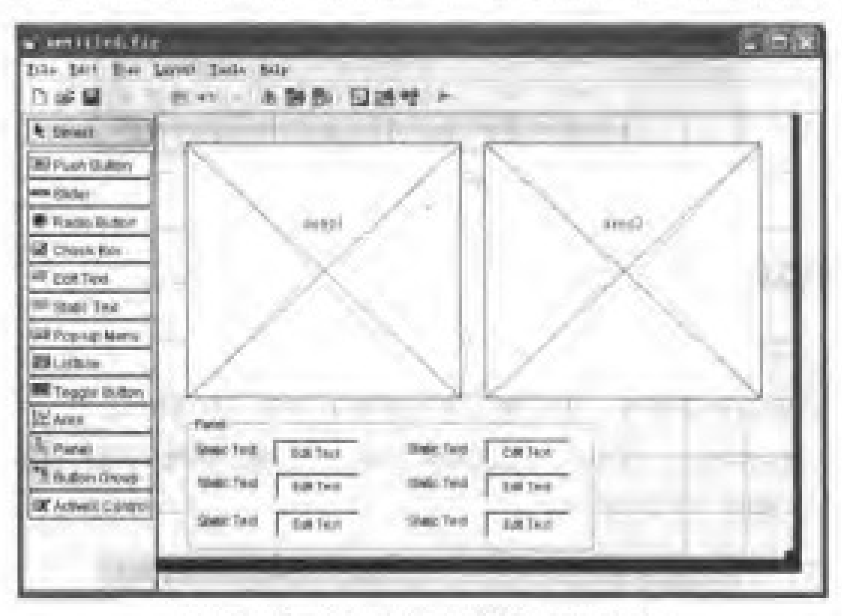


图11.18 复制编辑框和静态文本控件

step 8 添加"按钮（Push Button）"控件。选择控件面板中的"Push Button"对象，将其添加到面板中，如图11.19所示。

和其他编程实例类似，按钮是实现用户和程序互动的主要控件类型，通过鼠标单击按钮来实现某种行为并调用相应的回调子函数。

step 9 复制上面步骤的"按钮（Push Button）"控件。由于本实例中图形界面中包含三个按钮控件，因此在本步骤中需要复制该按钮控件，如图11.20所示。

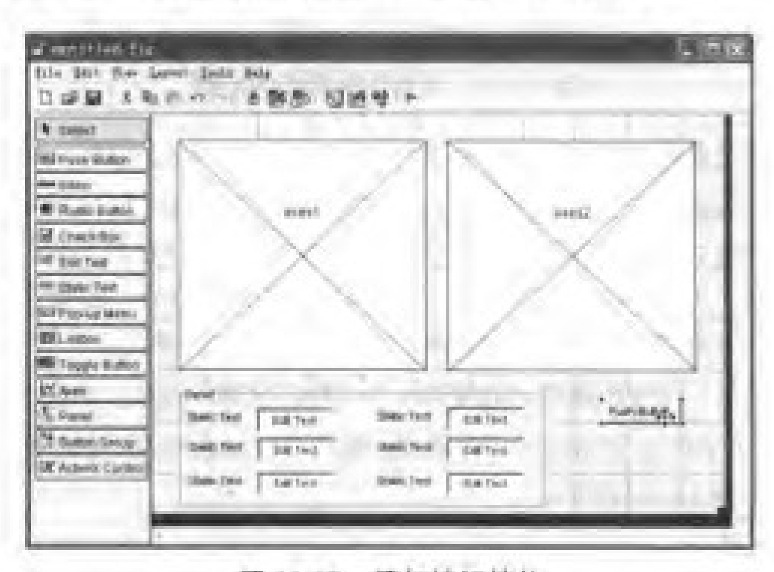


图11.19 添加按钮控件

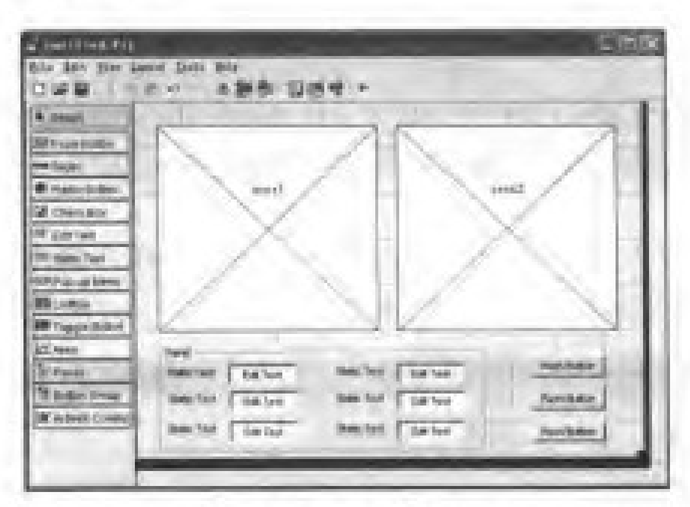


图 11.20 复制按钮控件

11.3.3 设置控件组件的属性

延续上面小节的步骤。

step 1 设置图形界面的标题属性。选中整个图形界面，单击 "Property Inspector" 按钮，打开 "Property Inspector" 对话框，然后选择 "Name" 选项，在其中输入标题 "3D Vector Field Simulator"，如图 11.21 所示。



图 11.21 设置图形界面的标题

step 2 设置组合框控件的属性。选中 "Panel" 控件，单击 "Property Inspector" 按钮，打开 "Property Inspector" 对话框，在其中设置组合框控件的属性，如图 11.22 所示。

在上面的 "Property Inspector" 对话框中，选择 "HighlightColor" 选项，将其设置为黑色，然后选择 "ShadowColor" 选项，将其设置为白色；最后，选择 "Title" 选项，将其设置为空白（也就是清除标题），其他属性保持系统的默认属性。

step 3 设置 "静态文本" 控件的属性。选择第一个 "静态文本" 控件，单击 "Property Inspector" 按钮，打开 "Property Inspector" 对话框，选择 "String" 选项，在其中输入 "X component"，得到的结果如图 11.23 所示。

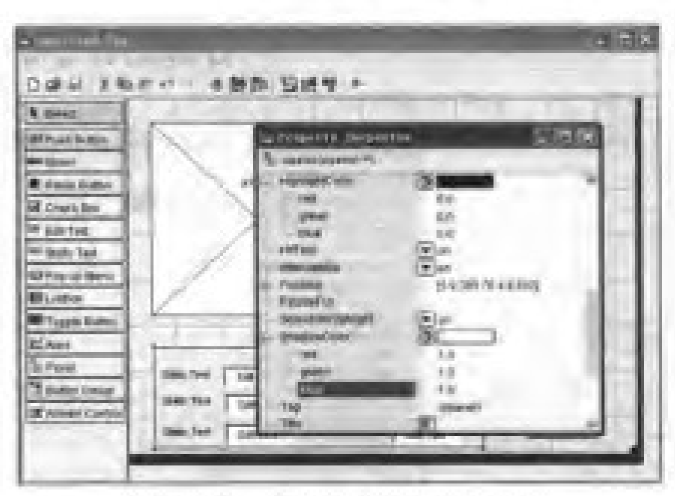


图 11.22 设置组合框控件的属性

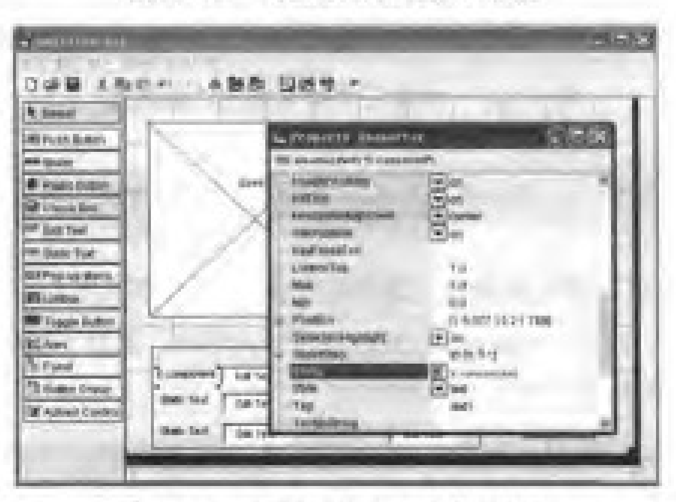


图 11.23 设置"静态文本"的标题

step 4 依次修改其他"静态文本"控件的文字。可以使用和上面步骤类似的方法来修改其他"静态文本"控件的文字，修改后的结果如图 11.24 所示。

step 5 设置左侧"编辑框"控件的属性。在本实例中，左侧的三个编辑框是提供给用户输入分量表达式，因此该编辑框的默认数值是"空格"。选择第一个"静态文本"控件，然后单击"Property Inspector"按钮，打开"Property Inspector"对话框，选择"String"选项，将其设置为"空"，得到的结果如图 11.25 所示。

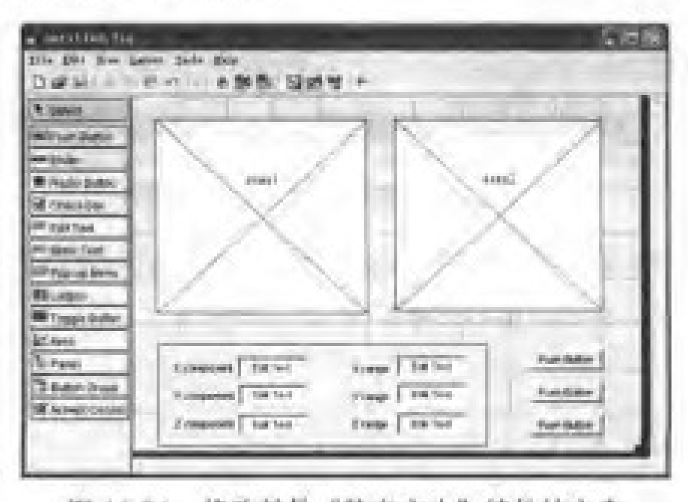


图 11.24 修改其他"静态文本"控件的文字

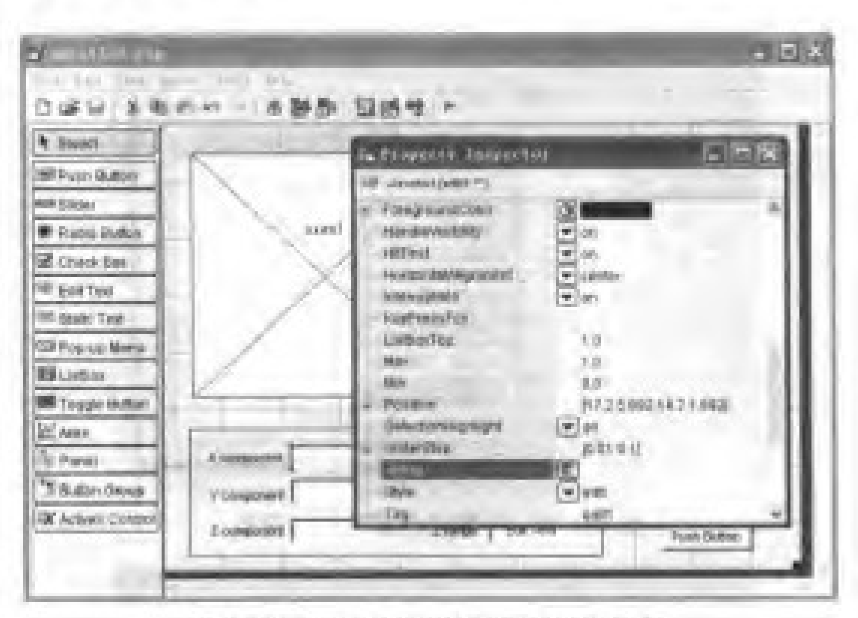


图 11.25　设置左侧编辑框控件的属性



对于左侧另外两个“编辑框”控件的属性，可以使用相同的方法进行设置。

step 6 设置右侧“编辑框”控件的属性。选择右侧第一个“静态文本”控件，单击“Property Inspector”按钮，打开“Property Inspector”对话框，选择“String”选项，将其设置为“[-1,1]”，得到的结果如图 11.26 所示。

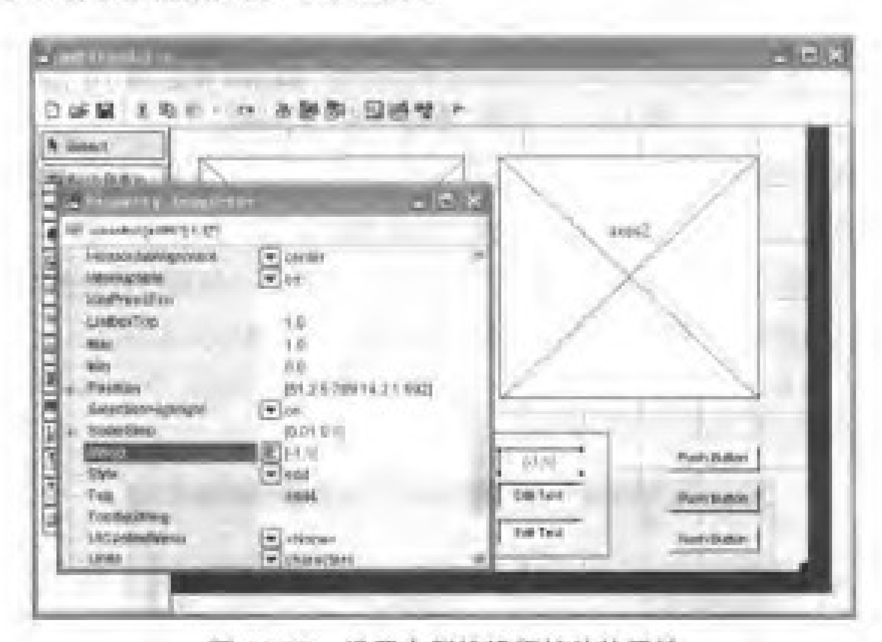


图 11.26　设置右侧编辑框控件的属性

step 7 查看编辑后的控件。使用相同的方法设置其他编辑框的属性，然后适当编辑其他编辑框的宽度，以及对齐方式，得到的最后结果如图 11.27 所示。

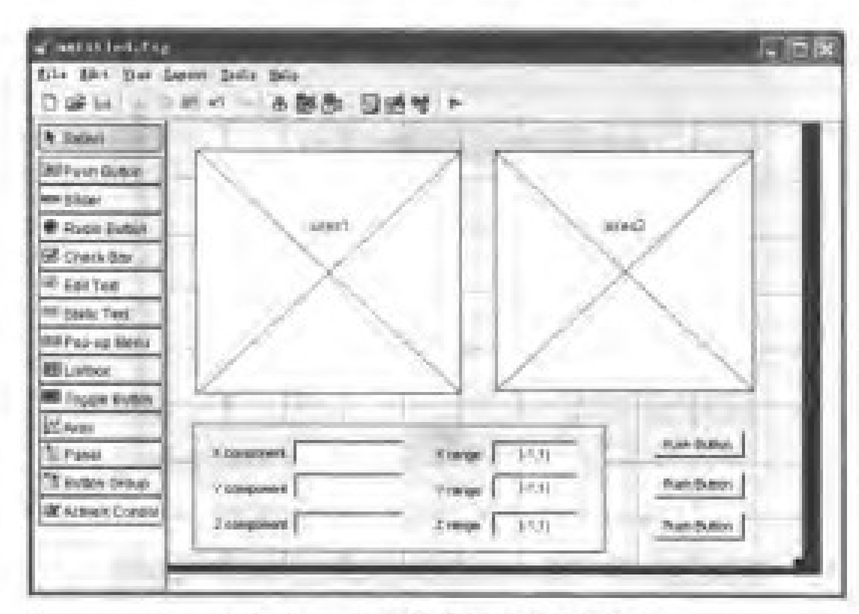


图 11.27　设置编辑框属性后的结果

step 8 设置“按钮”控件的名称。选择“按钮”控件，单击“Property Inspector”按钮，打开“Property Inspector”对话框，选择“String”选项，设置为“Draw”，如图 11.28 所示。

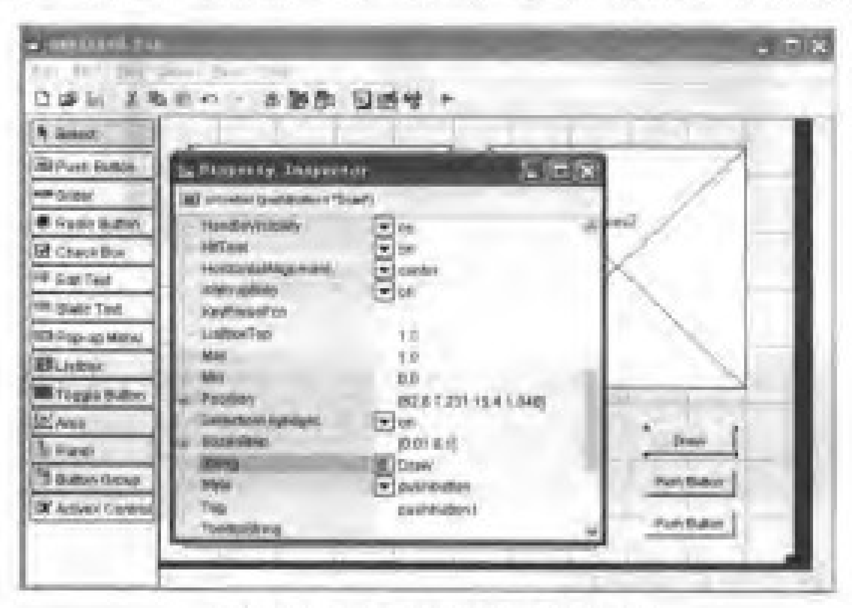


图 11.28　设置“按钮”控件的名称



依次设置另外两个“按钮”控件的名称，分别为“Output”和“Close”。设置的方法和上面步骤中的方法完全相同。

step 9 设置“按钮”控件的属性。选择“按钮”控件，单击“Property Inspector”按钮，打开“Property Inspector”对话框，选择“Enable”选项，选择为“Off”，如图 11.29 所示。

step 10 查看属性设置的结果。前面步骤已经设置控件所有的相关属性，选择 GUIDE 菜单栏中的“Tool”⇨“Run”命令，或者直接单击菜单栏中的“Run”按钮，查看属性设置的结果，如图 11.30 所示。



图 11.29　设置“按钮”控件的属性

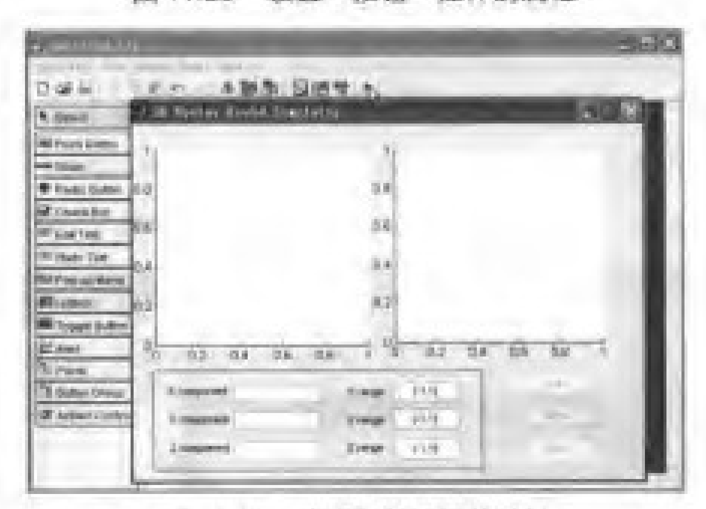


图 11.30　查看属性设置的结果



如果用户是第一次运行该命令，MATLAB 会提示保存该 FIG 文件，用户可以选择合适的保存路径和名称。在本例中，将其保存在用户的默认路径中，名称设置为 vectgui。

11.3.4　编写相应的程序代码

延续上面小节的步骤。

step 1　设置各控件的 "tag" 属性。选择 "Draw" 按钮，单击 "Property Inspector" 按钮，打开 "Property Inspector" 对话框，选择 "Tag" 选项，设置为 "Draw"，如图 11.31 所示。重复上面的步骤，为添加的各种控件重新设置 "Tag" 属性，设置的最后结果如下。

- 编辑框 Edit1：Tag 属性为 "X_Comp"；
- 编辑框 Edit2：Tag 属性为 "Y_Comp"；
- 编辑框 Edit3：Tag 属性为 "Z_Comp"；
- 编辑框 Edit4：Tag 属性为 "X_range"；
- 编辑框 Edit5：Tag 属性为 "Y_range"；
- 编辑框 Edit6：Tag 属性为 "Z_range"；
- 按钮 pushbutton2：Tag 属性为 "Output"；
- 按钮 pushbutton3：Tag 属性为 "Close"。



图 11.31 设置控件的"Tag"属性



之所以在本步骤中设置各控件的"Tag"属性，是为了方便在后面步骤中编写各对象的回调函数。例如，编写X_Comp的callback函数就比Edit1的callback函数更加直观而且容易理解。

step 2 打开M文件编辑器。选择GUIDE菜单栏中的"View"⇨"M-file Editor"命令，绘制直接单击菜单栏中的"M-file Editor"按钮，打开M文件编辑器，如图11.32所示。

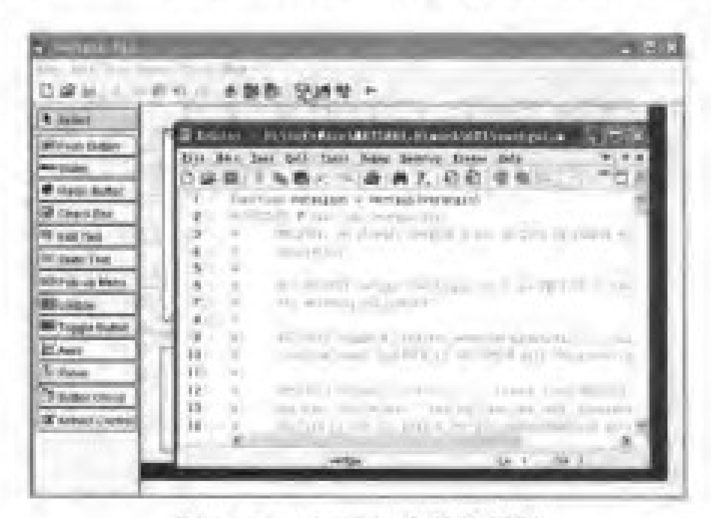


图 11.32 打开M文件编辑器

step 3 查看M文件程序代码。在前面介绍过，GUIDE会自动产生前面所有布局和属性设置的M文件。其部分代码程序如下：

```
function varargout = vectgui(varargin)
%   VECTGUI M-file for vectgui.fig
%      VECTGUI, by itself, creates a new VECTGUI or raises the existing
%      singleton*.
%      H = VECTGUI returns the handle to a new VECTGUI or the handle to
%      the existing singleton*.
%      VECTGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in VECTGUI.M with the given input arguments.
%      VECTGUI('Property','Value',...) creates a new VECTGUI or raises the
```

```
%       existing singleton*. Starting from the left, property value pairs are
%       applied to the GUI before vectgui_OpeningFunction gets called.  An
%       unrecognized property name or invalid value makes property application
%       stop.  All inputs are passed to vectgui_OpeningFcn via varargin.
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%        instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Copyright 2002-2003 The MathWorks, Inc.
% Edit the above text to modify the response to help vectgui
% Last Modified by GUIDE v2.5 17-Jul-2006 11:02:59
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @vectgui_OpeningFcn, ...
                   'gui_OutputFcn',  @vectgui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before vectgui is made visible.
function vectgui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to vectgui (see VARARGIN)
% Choose default command line output for vectgui
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes vectgui wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = vectgui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

function Y_Comp_Callback(hObject, eventdata, handles)
% hObject    handle to Y_Comp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of Y_Comp as text
%        str2double(get(hObject,'String')) returns contents of Y_Comp
as a double

% --- Executes during object creation, after setting all properties.
function Y_Comp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Y_Comp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
..............................................//限于篇幅，省略了部分代码
% --- Executes on button press in Close.
function Close_Callback(hObject, eventdata, handles)
% hObject    handle to Close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

上面的程序代码代表了使用GUIDE创建GUI对象的典型代码结构，下面详细介绍其中各组成部分的结构。

- **函数名称**：function varargout = vectgui(varargin)，MATLAB会根据用户保存的名称定义该GUI对象的主函数名称。在本实例中使用的名称为vectgui。
- **函数的注释文字**：从"% VECTGUI M-file for vectgui.fig"到"% Last Modified by GUIDE v2.5 17-Jul-2006 11:02:59"部分，是由MATLAB自动生成的注释文字。
- **GUI对象的初始化代码**：从"% Begin initialization code - DO NOT EDIT"到"% End initialization code - DO NOT EDIT"代码部分，是MATLAB自动生成的GUI对象初始化程序代码，这部分代码用户不能编辑（DO NOT EDIT）。
- **函数vectgui_OpeningFcn的程序代码**：该函数也是MATLAB自动设置的函数，该部分程序代码在GUI可见之前执行，也就是在打开GUI对象之前执行的程序代码。
- **函数vectgui_OutputFcn的程序代码**：该函数也是自动设置的函数，其功能是完成GUI对象的输出结果，这个程序代码将决定整个GUI对象的输出结果。
- **控件的创建函数**：控件的创建函数功能是用来创建控件并设置控件的属性，例如函数"Y_Comp_CreateFcn"的功能就是创建控件"Y_Comp"的程序代码。对于用户在GUIDE中添加的所有控件，MATLAB都会设置对应的创建函数。
- **控件的回调函数**：这部分函数将是编辑GUI的M文件的主要部分，主要功能就是编写互动事件的程序。例如，函数"Draw_Callback"的功能就是编写单击"Draw"按钮所响应的事件和行为。因此，回调函数是实现GUI互动功能的主要部分，也是用户编写代码的主要部分。



根据上面的分析可以得出，在后面的步骤中将主要添加各种控件的回调函数，实现各种不同的互动功能。

step 4 编写"X_Comp"控件的回调函数。为了方便用户编写回调函数，将M文件编辑器回嵌到MATLAB的命令窗口中，然后选择"X_Comp_Callback"选项，MATLAB会自动跳到程序代码的对应位置，如图11.33所示。

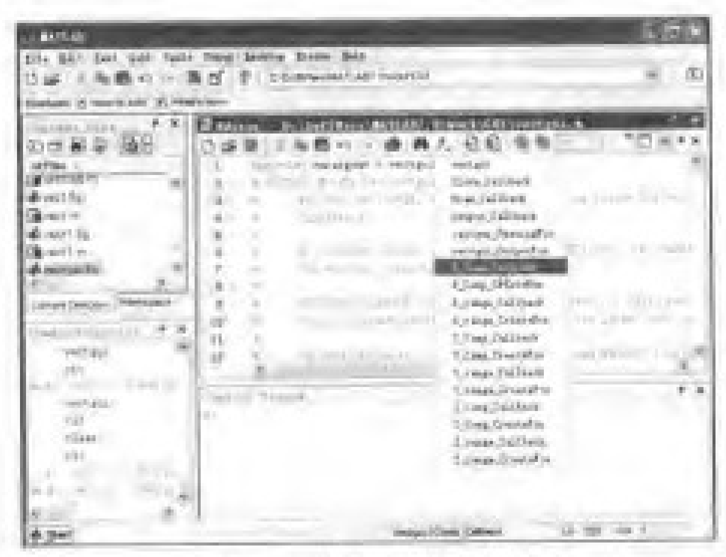


图 11.33　选择对应的程序代码

在MATLAB转到的位置上，编写对应的程序代码，详细代码如下：

```
function X_Comp_Callback(hObject, eventdata, handles)
% hObject    handle to X_Comp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of X_Comp as text
%        str2double(get(hObject,'String')) returns contents of X_Comp
as a double

set(handles.Draw,'Enable','On');          % Enable Draw_pushbutton after
function is available
guidata(hObject, handles);
```

在上面的程序代码中，只有后面两行代码是用户编写的，注释文字是MATLAB自动生成的，用户可以不用编写或者修改。上面两行代码的功能是：当用户在编辑框中输入相应的表达式后，将Draw按钮功能设置为可用，然后将编辑框中输入的数值传递到相应的变量中。



对于GUI对象中左侧的三个编辑框，用户可以编写对应的回调函数，其具体的程序代码和上面控件程序完全相同，这里就不重复介绍了。

step 5 编写"X_range"的回调函数。在MATLAB的命令窗口中，选择"X_range_Callback"选项，编写对应的程序代码，编写后的代码如下：

```
function X_range_Callback(hObject, eventdata, handles)
% hObject    handle to X_range (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of X_range as text
```

```
%        str2double(get(hObject,'String')) returns contents of X_range
as a double

guidata(hObject, handles);
```

上面程序代码的功能十分简单，就是将用户在 "X_range" 控件中输入的数值传递给对应的程序变量。对于其他的编辑框控件，用户可以添加相同的程序代码。

step 6 编写 "Draw" 按钮的回调函数。在MATLAB的命令窗口中，选择 "Draw_Callback" 选项，然后编写对应的程序代码，编写后的代码如下：

```
function Draw_Callback(hObject, eventdata, handles)
% hObject    handle to Draw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla;         % 清除图形窗口中的图形
x_range = str2num(get(handles.X_range,'String'));
y_range = str2num(get(handles.Y_range,'String'));
z_range = str2num(get(handles.Z_range,'String'));
x_comp = get(handles.X_Comp,'String');
y_comp = get(handles.Y_Comp,'String');
z_comp = get(handles.Z_Comp,'String');
%计算图形三维方向上的刻度范围
xmin = x_range(1);
xmax = x_range(2);
ymin = y_range(1);
ymax = y_range(2);
zmin = z_range(1);
zmax = z_range(2);
%计算三维方向上的刻度间隔
dx = (xmax-xmin)/4;
dy = (ymax-ymin)/4;
dz = (zmax-zmin)/4;
%判断间隔是否为0
if dx ~= 0
    xm = xmin:dx:xmax;
else
    xm = zeros(1,5);
end
    if dy ~= 0
    ym = ymin:dy:ymax;
else
    ym = zeros(1,5);
end
    if dz ~= 0
    zm = zmin:dz:zmax;
else
    zm = zeros(1,5);
end
%创建三维数据网格数据点
[meshx,meshy,meshz]=meshgrid(xm,ym,zm);
if x_comp ~= '0'
x_fun = inline(x_comp,'x','y','z');             % 创建x向函数的内联函数
    xc = x_fun(meshx,meshy,meshz);
else
```

```
    xc = zeros(5,5,5);
end
if y_comp ~= '0'
    y_fun = inline(y_comp,'x','y','z');                    %创建y向函数的内联函数
    yc = y_fun(meshx,meshy,meshz);
else
    yc = zeros(5,5,5);
end
if z_comp ~= '0'
    z_fun = inline(z_comp,'x','y','z');                    %创建z向函数的内联函数
    zc = z_fun(meshx,meshy,meshz);
else
    zc = zeros(5,5,5);
end
%设置绘制图形的坐标轴数值
axes(handles.field_axes)
%将坐标轴的句柄返回MATLAB的工作空间
handles.field_axes_handle = gca;
%绘制矢量图
quiver3(meshx,meshy,meshz,xc,yc,zc)
axis vis3d
%添加坐标轴名称
xlabel('x')
ylabel('y')
zlabel('z')
%添加网格线
grid on
%计算曲度数值
[curlx,curly,curlz]=curl(meshx,meshy,meshz,xc,yc,zc);
%设置图形的坐标轴属性，并将坐标轴的句柄返回工作空间
axes(handles.curl_axes)
handles.curl_axes_handle = gca;
%绘制矢量图
quiver3(meshx,meshy,meshz,curlx,curly,curlz)
axis vis3d
xlabel('x')
ylabel('y')
zlabel('z')
grid on
%计算数据的散度
div = divergence(meshx,meshy,meshz,xc,yc,zc);
%将坐标轴属性返回工作空间
handles.position = [meshx(:),meshy(:),meshz(:)];
handles.vector = [xc(:),yc(:),zc(:)];
handles.divergence = div(:);
handles.curl = [curlx(:),curly(:),curlz(:)];
set(handles.Output,'Enable','On');
guidata(hObject, handles);
```

在上面的程序代码中，首先将用户输入的参数赋给相应的程序变量，使用绘制命令将这些数据绘制成图形，并且添加图形标签、网格线等对象；最后，将所有计算结果和图形对象返回MATLAB的workspace中。



上面的程序代码是本GUI对象中最主要的功能代码，主要实现绘制图形的功能。

step 7 编写 "Output" 和 "Close" 按钮的回调函数。在MATLAB的命令窗口中，依次选择 "Output_Callback" 和 "Close_Callback" 选项，然后编写对应的程序代码：

```
% --- Executes on button press in Output.
function Output_Callback(hObject, eventdata, handles)
% hObject    handle to Output (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.Close,'Enable','On');       % enable close_pushbutton after
output
uiresume    % resume GUI

% --- Executes on button press in Close.
function Close_Callback(hObject, eventdata, handles)
% hObject    handle to Close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.figure1)
```

11.3.5 运行GUI对象

延续上面小节的步骤。

step 1 运行GUI对象。在MATLAB的命令窗口中输入 "vectgui"，然后按 "Enter" 键，MATLAB会自动调用前面步骤中编写的程序代码，用户可以输入相应的表达式和绘图范围，然后单击 "Draw" 按钮，查看绘制的结果，如图11.34 所示。

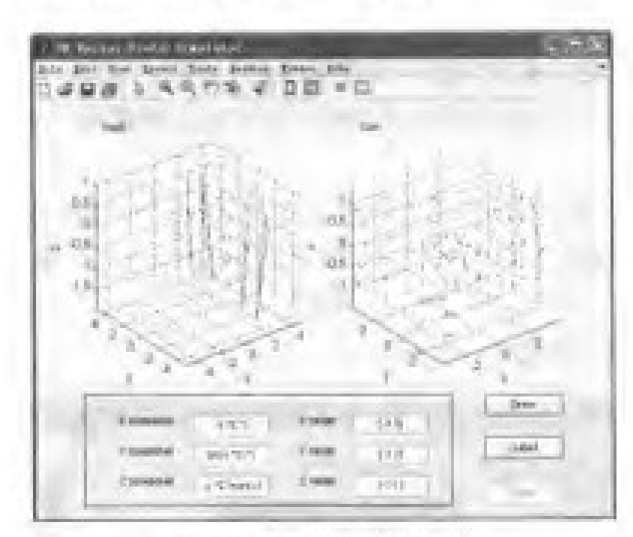


图11.34 运行GUI对象



说明 从上面的过程中可以看出，当用户在GUI界面中输入表达式和数值范围之后，"Draw" 按钮变得可用。当用户单击 "Draw" 按钮后，MATLAB会调用程序绘制图形，同时，"Output" 变得可用。

step 2 退出GUI对象。最后，当单击 "Output" 按钮后，系统会退出对应的程序运算，GUI对象中的 "Close" 按钮变得可用，如图11.35 所示。

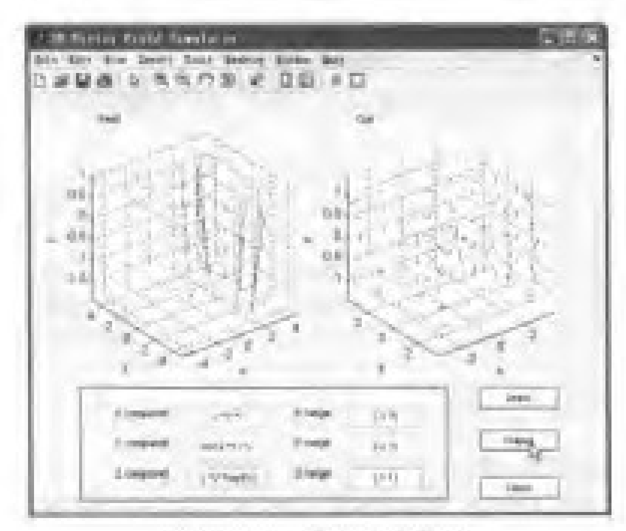


图 11.35　退出应用程序

当GUI用户界面中的"Close"按钮变得可用时，用户单击"Close"按钮后，就可以关闭GUI，同时退出相应的应用程序。



由于在前面的步骤中，用户分别创建了vectgui.fig和vectgui.m两个文件，为了能够有效地发布该GUI对象，建议将上面两个文件保存在同一个文件夹中。如果发布范围比较广，还可以添加一个文本文件，说明如何运行和使用该GUI对象。

11.3.6 GUIDE 创建 GUI 的注意事项

前面已经分步骤详细介绍了如何创建一个简单的GUI对象，在本小节中，将介绍使用GUIDE来创建GUI的一些主要注意事项。为了能够更加方便地介绍这些注意事项，还是沿用上面小节使用的GUI对象为例。

首先，在GUIDE中可以使用多种工具为多个控件对象进行精细排列，其中"排列对象（Align Objects）"对话框是比较常用的工具。以上面小节中的GUI对象为例，使用GUIDE打开上面小节中的GUI对象。然后同时选中三个"按钮"控件，选择菜单栏中的"Tools"⇨"Align Objects"命令，打开"Align Objects"对话框，如图11.36所示。



图 11.36　排列对象对话框

在上面的对话框中，可以选择多个排列方式，同时为每个排列方式设置具体的间距，单位是 pixels（像素），当选定了对应的排列方式和间距后，单击对话框中的"Apply"按钮，就可以查看修改的排列结果。



如果添加的是单个控件，而不是控件组，那么上面的对话框就很难有效果。如果使用鼠标来移动控件的单位移动量过大，可以选中控件后，使用方向键来移动控件，对控件的位置进行微调。

除了上面介绍的"Align Objects"对话框外，MATLAB 还提供"Grid and Rulers"对话框，可以在该对话框中设置网格和标线的属性。选择菜单栏中的"Tools"⇨"Grid and Rulers..."命令，打开"Grid and Rulers"对话框，如图 11.37 所示。



图 11.37　网格和标线对话框

在上面的对话框中，可以控制标线、网格的属性，设置网格间距等。在默认情况下，网格线的间隔是 50 像素，可以根据需要自行设置间距数值，MATLAB 接受的间距数值范围是 10～200 像素。当选中上面对话框中的"Show rulers"选项后，将会在 GUI 界面周围出现标尺。



在默认情况下，MATLAB 会选中"Snap to grid"选项。当用户在某个网格线周围 9 像素范围内移动或者重画对象，系统都会自动将该对象放置在网格线中。还需要提示用户，无论是否选中"Show grid"选项，选择"Snap to grid"选项都将有效。

在 MATLAB 的 GUIDE 中，除了提供了可以精细排列控件的工具之外，还提供了查看各种对象的"对象浏览器"对话框。选择 GUIDE 菜单栏中的"View"⇨"Object Browser"命令，打开"Object Browser"对话框，如图 11.38 所示。

在上面的对象浏览器中，用户可以查看各个图形对象的名称和继承关系，当 GUI 对象中包含多个控件时，在该对象浏览器中可以很方便地实现对象管理。



在本实例中，由于对象关系比较简单，因此在上面的对话框中没有出现层次展开按钮，用户可以选用其他比较复杂的 GUI 对象，查看对应的对象浏览器。



图 11.38 对象浏览器

在本小节的最后，将介绍设定GUI组态的对话框工具。选择GUIDE菜单栏中的"Tools"⇨"GUI Options"命令，打开"GUI Options"对话框，在该对话框中可以设置GUI初始状态的属性，如图11.39所示。

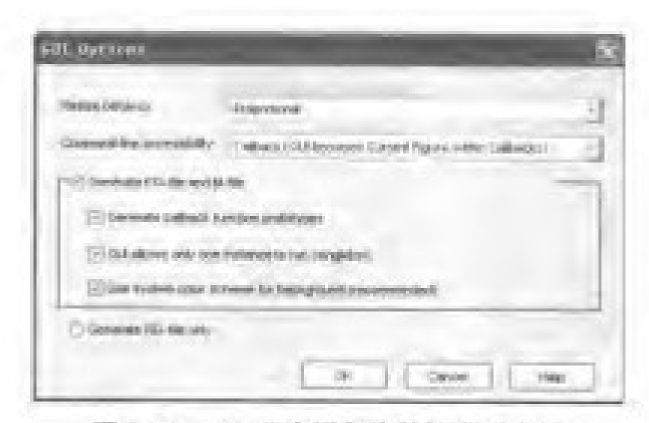


图 11.39 GUI应用程序的选项对话框

在创建GUI对象的时候，上面对话框中的各个属性都是十分重要的。为了帮助用户更好地理解GUI的各种属性，下面详细介绍对话框各个选项的具体含义。

11.3.7 重画行为（Resize behavior）

该选项的各个属性值表示MATLAB如何处理重画性质，MATLAB提供如下三个选项。

- **Non-resizable**：用户不能自行修改窗口的大小，这是MATLAB的默认数值。
- **Proportional**：该选项数值允许MATLAB按照新的图形窗口尺寸来重新绘制GUI控件，但是在重画过程中将不会改变标签中字号的大小。对于设置过程始终不关闭的简单GUI工具和对话框来讲，该选项是最佳的选择。
- **User-specified**：通过编程使重画过程中GUI按照用户指定的方式改变，选用该选项后，需要编写ResizeFcn属性定义的回调函数，该函数将根据新的图形窗口尺寸重新计算控件的大小和位置。

11.3.8 命令行访问（Command-Line Accessibility）

前面介绍过，当在MATLAB中创建图形对象时，各个图形对象的句柄可以通过各种函数来访问。当创建GUI对象时，也需要创建一个图形窗口；当图形窗口中包含坐标轴等对象时，需要编写命令行来访问该图形窗口。

在MATLAB中，可以为GUI设置如下三种命令访问权限。

- off：禁止命令对GUI图形窗口的访问。在这种情况下，GUI图形窗口的图形句柄是隐藏的。为此，应用程序M文件将创建一个对象句柄结构体来保存GUI中的所有用户控件句柄，并将该结构传递给子函数，保证GUI中句柄的准确使用。
- on：允许命令行对GUI图形窗口进行访问。
- User-specified：用户可以设置Handle Visibility和IntergerHandle两个图形窗口属性值，决定命令行是否能够访问图形窗口的句柄。Handle Visibility属性决定图形窗口的句柄对访问当前图形窗口的命令行是否可见，如果该属性值为off，则图形窗口的句柄从根对象的子对象列表中删除，因此该图形窗口将不是当前图形窗口，但是该图形窗口的句柄依然有效。IntergerHandle属性决定图形窗口的句柄是一个整数还是浮点数。如果该属性值为off，MATLAB将使用浮点数来代替整数。



生成FIG文件和M文件（Generate FIG-file and M-file）

如果希望通过使用GUIDE创建GUI对象的时候，同时生成FIG文件和应用程序M文件，则可以在该对话框中选中“Generate FIG-file and M-file”选项。选中该选项后，用户可以对其设置相应的属性选项，下面详细介绍。

- **生成回调函数原型（Generate Callback Function Prototypes）**：当选中该选项后，GUIDE将会在应用程序M文件中为每一个控件添加一个回调函数（需要提醒用户的是，组合框和静态文本控件不包含Callback属性）。用户可以自行添加回调函数的程序代码。
- **同一时间只允许运行一个应用程序实例（GUI Allows Only One Instance to Run（Singleton））**：选中该选项后，MATLAB在一次应用程序运行过程中只有一个GUI实例。如果GUI已经存在，MATLAB将该GUI带到前台，而不会重新创建一个新的GUI图形窗口。如果允许MATLAB显示多个GUI实例，则每一个调用命令都会创建一个新的图形窗口。
- **使用系统背景颜色设置（Using the System Background Colors）**：GUI控件所使用的颜色和计算机系统有关。如果选中该选项，则使图形窗口的背景颜色和用户添加的控件默认背景颜色相互匹配。



仅生成FIG文件（Generate FIG-file only）

如果不希望GUIDE生成对应的应用程序M文件，可以选中该选项。当用户在界面设计编辑器中保存GUI对象时，GUIDE将仅仅创建FIG文件，用户可以使用open命令或者hgload命令来显示该文件。当希望创建一个与应用程序M文件完全不同的实例时，可以选中该选项。



定制标准菜单

在GUI控件对象中，界面菜单（uimenu）是一个重要的组成部分。从句柄图形对象结构中，Uimenu对象的结构体系以Figure图形窗口为父对象，和Axes坐标轴、Unicontrol界面控件为平等级别的组件。

在MATLAB中，可以根据需要在GUI对象中创建标准菜单，自行设置菜单或者创建现场菜单等。同时，可以设置菜单控件的各种属性，例如添加快捷键、设置对应的回调函数等。因此，在GUI对象中，可以设置菜单控件来完成多种功能。对于比较简单的GUI对象，可以根据需要定制MATLAB图形



窗口的标准菜单，设置不同的菜单属性。下面介绍一个简单的实例，来说明如何定制标准菜单。

例 11.3 创建一个简单的GUI对象，根据需要定制标准菜单。

step 1 创建一个默认的图形窗口。在MATLAB的命令窗口中输入下面的命令代码：

```
>> Handle_figure=figure;
```

step 2 查看图形结果。输入上面的代码后，按"Enter"键，得到默认的图形窗口，如图11.40所示。



图 11.40 创建默认图形窗口

在默认情况下，MATLAB的图形窗口有一个顶层菜单条（Top-Level Menu）。在MATLAB7.0中，该菜单条包括8个菜单项目：File（文件）、Edit（编辑）、View（视图）、Insert（插入）、Tools（工具）、Desktop（桌面）、Window（窗口）和Help（帮助）。当用户单击每一个菜单项时，都会产生对应的下拉菜单（Pull-Down Menu）。



在上面的菜单项中，Desktop（桌面）选项是在MATLAB7.0中添加的新菜单选项，在MATLAB6.x版本中没有该菜单选项。

step 3 查看Menu属性列表。在命令窗口中输入命令"set(Handle_figure,'MenuBar')"，得到的结果如下：

```
>> set(Handle_figure,'MenuBar')
[ none | {figure} ]
```

从上面的结果中可以看出，标准菜单"MenuBar"有两个属性[none | {figure}]。当对象的属性选取none时，表示图形窗口不显示标准菜单以及对应的工具条；当对象的属性选取"figure"属性时，图形窗口将显示标准菜单，这是图形窗口菜单的默认属性。

step 4 隐藏标准菜单。在有些程序项目中，为了不让用户对程序对象进行操作，可以隐藏菜单。在命令窗口中输入下面的代码：

```
>> set(Handle_figure,'MenuBar',menubar);
```

step 5 查看图形结果。输入代码后，按"Enter"键，得到的结果如图11.41所示。



除了上面的命令之外，还可以通过命令set(Handle_figure,'MenuBar','none')来隐藏图形窗口的菜单选项。



图 11.41 隐藏菜单选项后的图形窗口

step 6 恢复标准菜单。在命令窗口中输入下面的代码：

```
>> set(Handle_figure,'MenuBar','figure')
```

step 7 查看图形结果。输入代码后，按"Enter"键，得到的结果如图 11.42 所示。

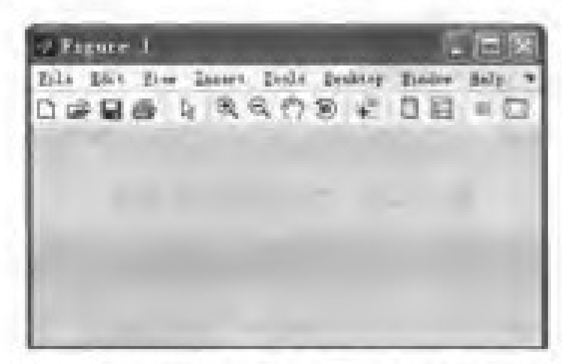


图 11.42 恢复菜单后的图形窗口

11.5 使用 GUIDE 创建自定义菜单

在前面介绍过，用户可以使用 GUIDE 或者 M 文件来创建 GUI 对象。在本小节中，将选用一个实例来介绍如何使用 GUIDE 创建菜单对象，然后实现各种功能。由于该实例比较复杂，因此在本节中将分小节详细介绍。

11.5.1 创建图形界面

例 11.4 创建一个 GUI 对象，用户可以在该图形界面中实现图像转换等功能，其完成的最后结果如图 11.43 所示。

从图中的结果中可以看出，该图形窗口中的菜单选项是用户自定义的，而不是 MATLAB 默认的标注菜单。在本实例中，菜单选项并不复杂，只有 File 和 Thresholding Method 两个菜单选项条，而且两个菜单选项都分别有一些子菜单选项，用以实现各种功能。

step 1 打开"Menu Editor"对话框。使用前面小节所使用的方法打开一个默认的空白 GUI 设计面板，然后选择该编辑界面的菜单栏中的"Tools"⇨"Menu Editor"命令，打开"Menu Editor"对话框，如图 11.44 所示。

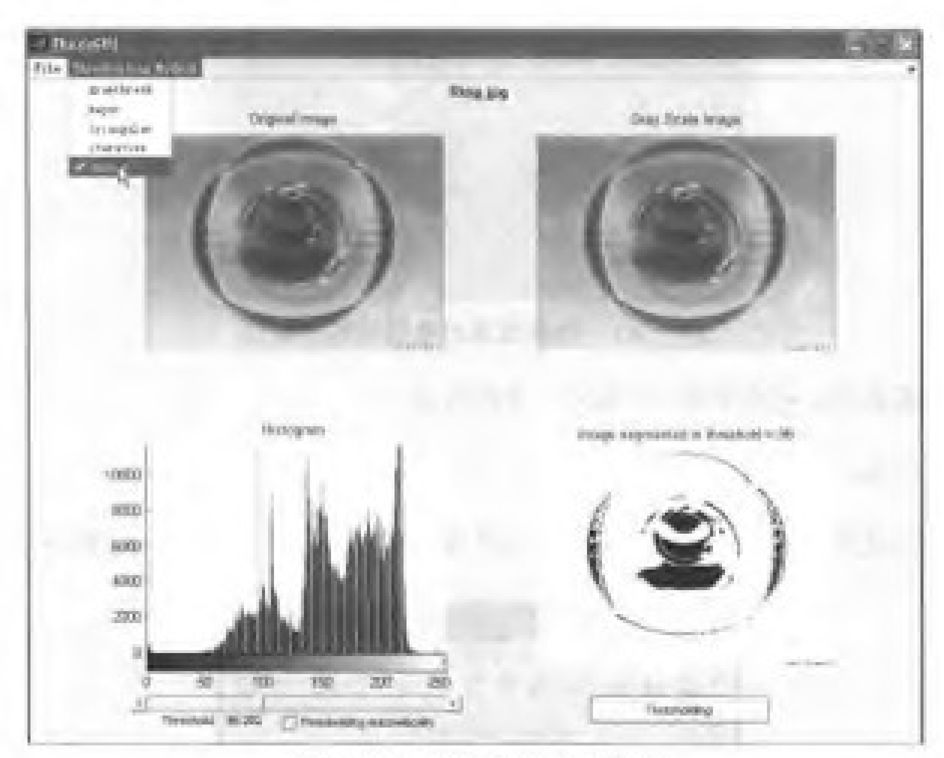


图 11.43　完成后的 GUI 界面



图 11.44　菜单编辑器的外观

"Menu Editor" 对话框是用户使用GUIDE 创建菜单控件的主要操作界面，用户可以在该对话框中添加和设置菜单控件的属性。



一般情况下，GUIDE 可以创建两种类型的菜单：在图形窗口的菜单栏中显示的菜单栏菜单（Menu Bar），当用户在图形窗口对象上单击右键时弹出的现场菜单（Context Menus）。

step 2 创建第一个菜单栏菜单。单击 "Menu Editor" 对话框中的 "New Menu" 按钮，创建一个新的菜单栏菜单，然后在对话框右侧的列表中设置其菜单属性，如图 11.45 所示。

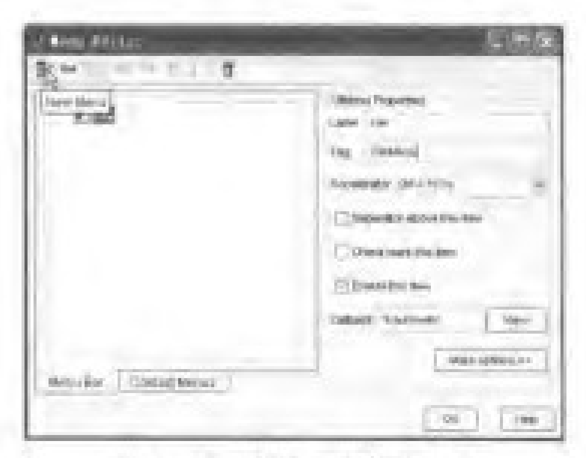


图 11.45　创建一个菜单选项

在上面的菜单选项中，Label选项就是菜单在GUI图形界面中的名称"File"。Tag属性选项就是用户在编写对应的应用程序中的引用名称"FileMenu"。



在默认情况下，MATLAB会选中"Menu Editor"对话框中的"Enable this item"选项，表示当GUI图形界面在启动的时候，该菜单选项是可用的。如果希望在启动GUI图形界面时禁用菜单，就需要取消选中该选项。

step 3 添加第一个子菜单选项。单击"Menu Editor"对话框中的"New Menu Item"按钮，添加第一个子菜单选项。然后在对话框右侧的列表中设置其菜单属性，如图11.46所示。



图 11.46　添加第一个子菜单选项

这个步骤中添加的子菜单选项，就是在GUI图形界面中"File"菜单中的下拉菜单选项，该菜单选项的Label是"Open Image"，Tag属性是"OpenMenuItem"，其他选项则保持系统的默认设置。

step 4 依次添加"File"菜单的其他子菜单选项。重复步骤3的做法，依次为"File"菜单添加其他子菜单选项，如图11.47所示。

上面步骤中添加的两个子菜单选项如下。

- **Label**：Save Image；　**Tag**：Save。
- **Label**：Close；**Tag**：CloseMenuItem。

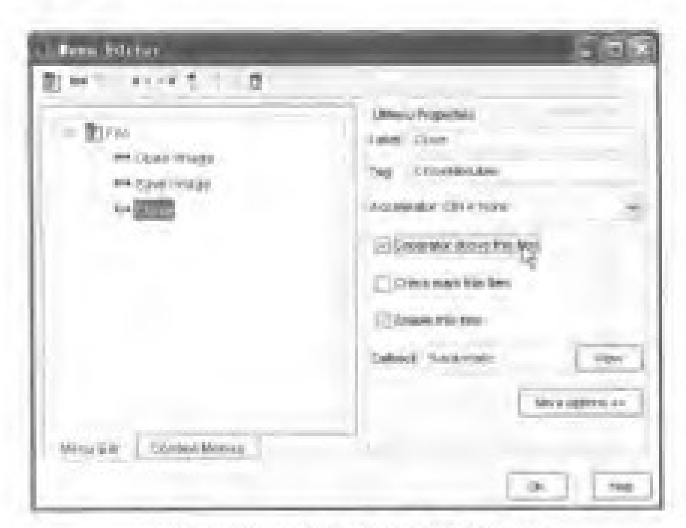


图 11.47　添加其他子菜单选项

对于最后一个子菜单选项，为了将其和上面的子菜单选项隔开，这样前面两个子菜单选项就被分成一组，在其属性列表中选中了"Separator above this item"，MATLAB 会在该菜单选项上方添加一个分隔线。



前面步骤已经完成了第一个菜单选项的设置过程，主要涉及的内容是设置菜单选项的外观属性。但是在"Menu Editor"对话框中，除了可以设置外观属性之外，还可以设置其他菜单属性，包括回调函数 Callback 等。

step 5 查看前面步骤设置的结果。单击"Menu Editor"对话框中的"OK"按钮，保存上面步骤中的所有设置。然后选择空白 GUI 面板中的"Tools" ⇨ "Run"命令，查看设置的结果，如图 11.48 所示。



图 11.48　前面步骤设置的结果



从上面的结果中可以看出，前面的步骤已经完成"File"菜单选项的所有子菜单选项的外观属性。可见，使用 GUIDE 来创建 GUI 的菜单选项步骤十分简单有效。

step 6 重新打开"Menu Editor"对话框。关闭上面的演示界面，然后返回到 GUI 的设计界面中，重新打开"Menu Editor"对话框，如图 11.49 所示。

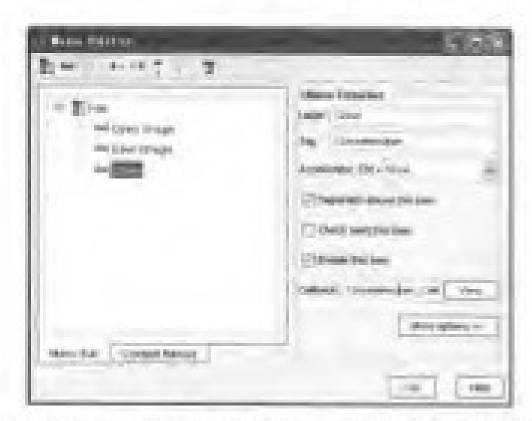


图 11.49　重新打开“Menu Editor”对话框



由于在前面的步骤中，用户已经运行了前面步骤的设置，MATLAB会为每个菜单选项设置相应的回调函数，因此，在菜单选项的“Callback”选项中会出现对应的函数名称，单击旁边的“View”按钮，就可以查看M文件中的对应函数。

step 7 添加新的菜单选项。使用和前面步骤类似的方法，用户可以依次添加其他的菜单选项，如图11.50所示。

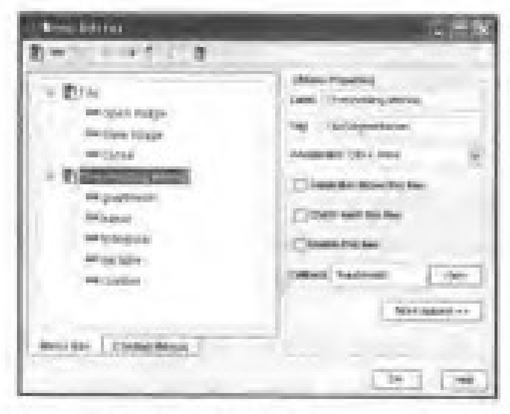


图 11.50　添加新的菜单选项



根据整个应用程序的需要，在上面的步骤中，将“Thresholding Method”菜单选项设置为不可用，因此当用户运行GUI对象时，该菜单选项将不可用。

step 8 查看新的菜单设置结果。单击“Menu Editor”对话框中的“OK”按钮，保存所有设置。然后选择GUI面板中的“Tools”⇨“Run”命令，查看设置的结果，如图11.51所示。



由于在前面步骤中将“Thresholding Method”菜单选项设置为不可用，因此在启动GUI时，将不能查看该菜单选项。

11.5.2 使用图形界面工具栏

延续上面小节的步骤。



图 11.51　查看菜单设置的结果

step 1 重新查看 "Menu Editor" 对话框。前面步骤设置了 GUI 对象中的所有菜单选项，并且查看了运行的结果，现在介绍 "Menu Editor" 对话框中的常见工具，如图 11.52 所示。



图 11.52　菜单设置对话框

step 2 查看对话框中的工具栏。在 "Menu Editor" 对话框中，用户可以使用菜单栏中的各种按钮，修改各种菜单选项的属性，具体的工具栏如图 11.53 所示。



图 11.53　工具栏按钮

将上面的工具栏按钮依次命名为 1～8，下面详细介绍各按钮的功能：

- **按钮 1：** 创建新的菜单（New Menu）。当单击该按钮后，MATLAB 就会创建一个新的菜单栏菜单选项。
- **按钮 2：** 添加新的子菜单选项（New Menu Item）。当单击该按钮后，MATLAB 就会在当前菜单选项下面添加子菜单选项。该按钮只有在添加菜单后才能使用。
- **按钮 3：** 添加新的现场菜单（New Context Menu）。当单击该按钮后，就会添加新的现场菜单。当选择 "Menu Bar" 选项卡时，该按钮不可用；只有当选择 "Context Menu" 选项卡时，才可以使用该按钮。
- **按钮 4：** 将菜单移到上一层（Move Selected Item Backward）。该按钮只能用在子菜单选项中，当用户单击该按钮后，子菜单选项会被移到上层，变成菜单栏菜单。

- **按钮5**：将菜单移到下一层（Move Selected Item Forward）。该按钮也只能适用于子菜单选项，当用户单击该按钮后，子菜单选项会被移到下一层，变成更下层的菜单选项。
- **按钮6**：将菜单选项向上移动一个单位（Move Selected Item Up）。该按钮也只能适用于子菜单选项，当用户单击该按钮后，子菜单选项会向上移动一个单位。
- **按钮7**：将菜单选项向下移动一个单位（Move Selected Item Down）。该按钮也只能适用于子菜单选项，当用户单击该按钮后，子菜单选项会向下移动一个单位。
- **按钮8**：删除菜单选项（Delete Selected Item）。当单击该按钮后，就可以删除所选中的菜单选项。

除了可以使用上面的按钮来设置各种菜单属性之外，对于比较详细的属性设置，可以单击"Menu Editor"对话框中的"More options>>"按钮，打开所选菜单选项的属性对话框，如图 11.54 所示。

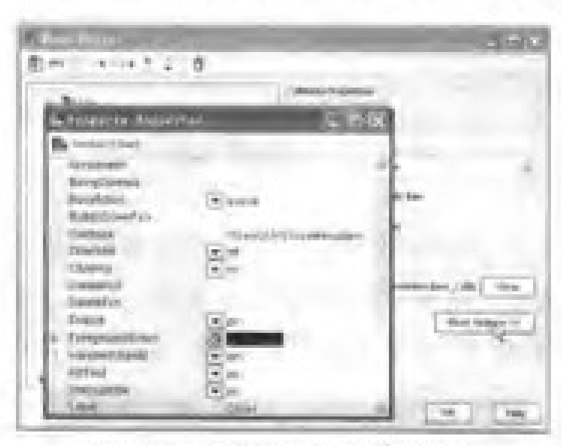


图 11.54 设置菜单选项的属性



在上面打开的"Property Inspector"对话框中，可以设置关于菜单的所有相关选项，相当于前面章节中设置其他控件的属性一样。

11.5.3 添加图形界面的控件

延续上面小节的步骤。

step 1 添加控件。根据本小节的要求，添加后的结果如图 11.55 所示。

step 2 分析控件的功能。添加各种控件的具体步骤，在这里就不详细介绍了。下面详细介绍各种控件的类型、功能和重要属性。

- 控件类型 Axes（坐标轴），"Tag"属性为"AxesImagen"，"Visible"属性为"Off"，功能是显示各种图像。
- 控件类型 Slider（滚动条），"Tag"属性为"Slider1"，"Visible"属性为"Off"，功能是调整图像转换的参数数值。
- 控件类型 Checkbox（复选框），"String"属性为"Thresholding automatically"，"Tag"属性为"ActualAutomatico"，"Visible"属性为"Off"，功能是提供自动转换功能。
- 控件类型 Pushbutton（按钮），"String"属性为"Thresholding"，"Tag"属性为"Umbralizar"，"Visible"属性为"Off"，功能是实现图形转换功能。

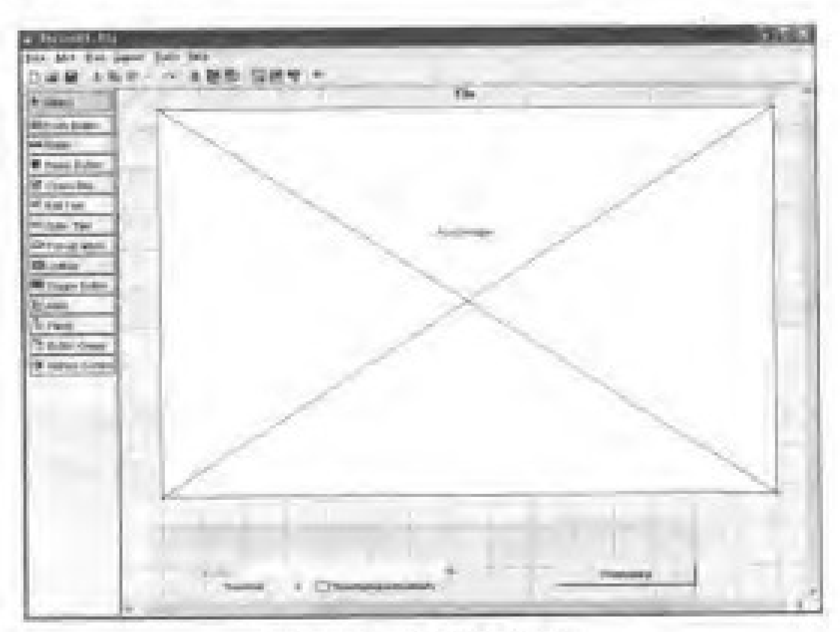


图 11.55　添加图形控件

step 3 查看对象浏览器。除了上面的控件类型之外，在上面的图形界面中还有一些"静态文本"（Static Text）控件类型，用来作为各种控件的标识。为了方便用户了解 GUI 对象中各控件的结构类型，可以查看该 GUI 对象的对象浏览器，如图 11.56 所示。



图 11.56　查看 GUI 的对象结构



从上面的对象结构中可以看出，该 GUI 界面的所有控件的 Visible 属性都是 Off，因此在用户加载图像文件之前，所有控件都是不可见的。

11.5.4 添加"File"菜单的回调函数

延续上面小节的步骤。

step 1 添加"Open Image"菜单选项的回调函数。回到"Menu Editor"对话框中，选择"Open Image"选项，单击"Callback"选框后的"View"按钮，如图 11.57 所示。

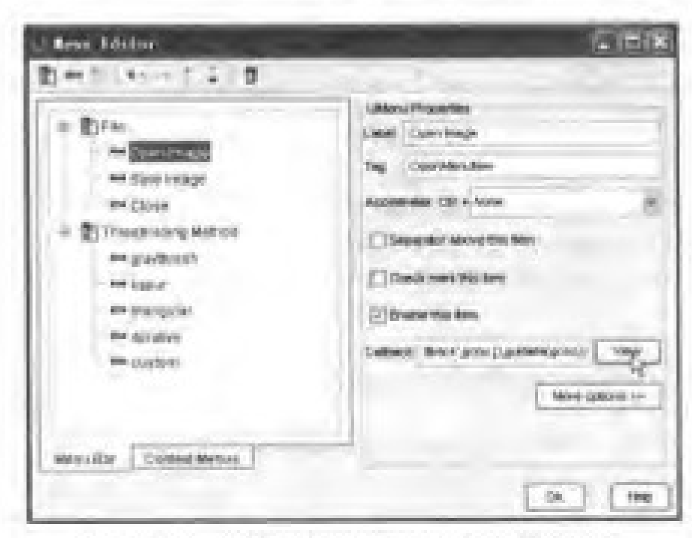


图 11.57　编写 "Open Image" 的回调函数

step 2 编写回调函数代码。当单击 "View" 按钮后，MATLAB 就会切换到 M 文件编辑器中对应的函数部分，用户可以在其中编写相应的程序代码，具体代码如下：

```
function OpenMenuItem_Callback(hObject, eventdata, handles)
% 定义全局变量
global Archivo NameArchivo

% 将 save 菜单选项设置为不可用
set(handles.Save,'enable','off')
set(handles.TipoSegmentacion,'enable','off')

% 打开图像文件
[NameArchivo,PathArchivo] = uigetfile({'*.jpg;*.tif;*.gif;*.bmp;*.png;
*.hdf;*.pcx;*.xwd;*.ico;*.cur;*.ras;*.pbm;*.pgm;*.ppm'});

% 如果读取图像文件操作正确，则显示文件，并将各种菜单选项设置为可用
if ~isequal(NameArchivo, 0)
    Archivo=[PathArchivo,NameArchivo];
    CargarImagen(handles)
    set(handles.Save,'enable','on')
    set(handles.TipoSegmentacion,'enable','on')
    set(handles.DefinidoUsuario,'checked','on')
end
```



上面的程序代码中，通过程序uigetfile语句来打开图像文件，然后分别设置对应图形窗口中的菜单选项的属性。

step 3 添加 "Save Image" 菜单选项的回调函数。回到 "Menu Editor" 对话框中，选择 "Save Image" 选项，然后单击 "Callback" 选框后的 "View" 按钮，在 M 文件编辑器对应的位置编写保存文件的代码如下：

```
function Save_Callback(hObject, eventdata, handles)
% 定义全部变量
global ImagenUmbral
% 如果在图形界面中没有加载图形文件，则显示错误信息
if isempty(ImagenUmbral)==1,msgbox('Doesn''t exist an image');return,
```

```
end
%显示保存文件的对话框
[filename,pathname] = uiputfile({'*.jpg';'*.tif';'*.gif';'*.bmp';'*.png';...
    '*.hdf';'*.pcx';'*.xwd';'*.ico';'*.cur';'*.ras';...
    '*.pbm';'*.pgm';'*.ppm'},'Save file name');
%如果不存在该文件，或者不存在保存路径，则显示错误信息
if isequal(filename,0) | isequal(pathname,0)
   errordlg('Saving canceled','Threshold GUI'); error('Saving canceled')
else
    try
        imwrite(ImagenUmbral,[pathname,filename]);
    catch
        errordlg('Error during saving','Threshold GUI'),error('Error
during saving')
    end %try
end %if
```



在上面的程序代码中，根据不同的保存文件和情况，显示不同的提示信息。只有当保存文件符合系统的要求，才会保存相应的文件。

step 4 添加"Close"菜单选项的回调函数。回到"Menu Editor"对话框中，选择"Close"菜单选项，然后单击"Callback"选框后的"View"按钮，在M文件编辑器对应的位置编写保存文件的代码如下：

```
% -----------------------------------------------------------------------
function CloseMenuItem_Callback(hObject, eventdata, handles)
% -----------------------------------------------------------------------
selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                     ['Close ' get(handles.figure1,'Name') '...'],...
                     'Yes','No','Yes');
if strcmp(selection,'No')
    return;
end
delete(handles.figure1)
```



当选择菜单选项中的"Close"选项时，会显示询问对话框，当用户选择对话框中的"Yes"按钮时，关闭整个对话框；如果选择对话框中的"No"按钮，则返回程序。

11.5.5 添加"Thresholding Method"菜单的回调函数

延续上面小节的步骤。

step 1 添加"Thresholding Method"菜单的回调函数。在对话框中选中对应的菜单选项，然后单击右侧的"View"，添加相应的函数代码，如图11.58所示。

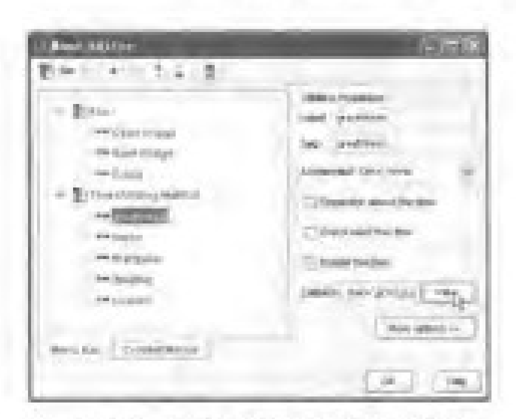


图 11.58 添加菜单选项的程序代码

step 2 添加具体的程序代码。所有子菜单选项对应的回调函数代码程序如下：

```
% --------------------------------------------------------------------
function graythresh_Callback(hObject, eventdata, handles)
% --------------------------------------------------------------------
global ImagenGris
%将所有菜单选项的属性设置为不选中
NoChecked(handles)
%将 "graythresh" 菜单选项的属性设置选中
set(handles.graythresh,'checked','on')
%调用 graythresh 图形转换函数
Umbral = graythresh(ImagenGris)*255;
GraficarHistoUmbral(Umbral,handles,'all')
% --------------------------------------------------------------------
function kapur_Callback(hObject, eventdata, handles)
% --------------------------------------------------------------------
global ImagenGris
%将所有菜单选项的属性设置为不选中
NoChecked(handles)
%将 "kapur" 菜单选项的属性设置选中
set(handles.kapur,'checked','on')
%调用 Kapur1 图形转换函数
Umbral = Kapur1(ImagenGris)*255;
GraficarHistoUmbral(Umbral,handles,'all')
% --------------------------------------------------------------------
function triangular_Callback(hObject, eventdata, handles)
% --------------------------------------------------------------------
global ImagenGris
%将所有菜单选项的属性设置为不选中
NoChecked(handles)
%将 "triangular" 菜单选项的属性设置为选中
set(handles.triangular,'checked','on')
%调用 triangular 图形转换函数
Umbrales = Triangular1(ImagenGris)*255;
set(handles.triangular,'userdata',Umbrales)
GraficarHistoUmbral(Umbrales(1),handles,'trian')
% --------------------------------------------------------------------
function iterativo_Callback(hObject, eventdata, handles)
% --------------------------------------------------------------------
global ImagenGris
%将所有菜单选项的属性设置为不选中
NoChecked(handles)
```

```
%将 "iterativo" 菜单选项的属性设置为选中
set(handles.iterativo,'checked','on')
%调用 iterativo 图形转换函数
Umbral = Iterativol(ImagenGris)*255;
GraficarHistoUmbral(Umbral,handles,'all')
% ---------------------------------------------------------------------
function DefinidoUsuario_Callback(hObject, eventdata, handles)
% ---------------------------------------------------------------------
%将所有菜单选项的属性设置为不选中
NoChecked(handles)
%将 "DefinidoUsuario" 菜单选项的属性设置为选中
set(handles.DefinidoUsuario,'checked','on')
set(handles.Umbralizar,'enable','on')
set(handles.Umbralizar,'string','Thresholding')
if get(handles.ActualAutomatico,'value')==1
    set(handles.Umbralizar,'enable','off')
end %if
```

step 3 查看图形转换函数的代码。在上面的程序代码中，对不同的菜单选项，调用不同的图形转换函数，这些函数都是用户自行定义的函数，对应的函数代码如下：

```
%=====================================================================
function umbral = Kapurl(Imagen)
%=====================================================================
%检测图像的属性
if ~Esgray(Imagen)
   uiwait(msgbox('La Imagen no se encuentra en escala de grises','Error',
'modal'))
    umbral = 0;
else
[ fil col] = size(Imagen);
%计算图像的像素数值
Pixeles = fil * col;
%计算图像的直方图像素值
    hl = imhist(Imagen);
Pi = hl/Pixeles;
Pt = zeros(256,1);
    Pt(1) = Pi(1);
    for i = 2:256
        Pt(i)=Pt(i-1)+Pi(i);
    end
%创建循环进行图像转换
    Hb = zeros(1,256);                              Hw = zeros(1,256);
    for i = 1:256
        if Pt(i) > 0
            for j = 1 : i
                if Pi(j) > 0
                    Hb(i) = Hb(i) + ((Pi(j) / Pt(i)) * log(Pi(j) / Pt(i)));
                end
            end
        end
    end
    for i = 1:256
        if (1-Pt(i)) > 0
            for j = i + 1 : 256
```

```
                if Pi(j) > 0
                    Hw(i) = Hw(i) + ((Pi(j) / (1-Pt(i))) * log(Pi(j) / (1-Pt(i))));
                end
            end
        end
    end
    Hb = -Hb;
    Hw = -Hw;
    H = Hb + Hw;
    [a, b] = max(H(:));
    umbral = b-1;
    umbral = umbral/255;
end

%==========================================================================
function Umbral = Triangular1(Imagen);
%==========================================================================
% 检测图像的属性
if ~Esgray(Imagen)                                         %Validar si una
Imagen se encuentra en Escala de Grises o en color
     uitwait(msgbox('La imagen no se encuentra en escala de grises',
'Error','modal'))
    Umbral = 0;
else
z = 0;
% 返回图像的直方图数据
H = imhist(Imagen);
% 调用 Promedio 函数，返回新的图像句柄
H = Promedio(H);
% 调用 Derivada 函数，返回 dh 数值
dh = Derivada(H);
% 调用 MaxiMini 函数，返回 dh 的最小值和最大值
[maxim minim] = MaxiMini(dh);
% 使用循环进行图像转换
    for i = 1 : length(maxim)-1;
        clear D, k = 0;
        x1 = maxim(i); y1 = H(maxim(i));
        x2 = maxim(i + 1); y2 = H(maxim(i + 1));
        M = (y2 - y1) / (x2 - x1);
        if (x2 - 1) - (x1 + 1) > 0
            for j = x1 : x2;
                Px = j;
                Py = H(j);
                k = k + 1;
               D(k,1) = sqrt(((Px - x1) * M - Py + y1)^2 / (M^2 + 1));
                D(k,2) = j;
            end
            [Pu p] = max(D(:,1));
            if Pu > 0
                z = z + 1;
                Umbral(z) = D(p,2);
            end
        end
    end
    Umbral = Repetidos(Umbral,10)/255;
end
```

```
% Promedio 子函数的子程序代码
function x = Promedio(h);
n = 5;
% 创建零值矩阵
g = zeros(256 + 2 * round(n / 2 - 1), 1);
% 将图像的直方图数据添加到矩阵 g 中
g(1 + round(n / 2 - 1) : length(g) - round(n / 2 - 1)) = h;
% 利用循环实现图像转换
for i = 1 + round(n / 2 - 1) : length(g)-round(n / 2 - 1);
    sum = 0;
    for j = i - round(n / 2 - 1) : i + round(n / 2 - 1)
        sum = sum + g(j);
    end
    x(i - round(n / 2 - 1), 1) = round(sum / n);
end
% Derivada 子函数的子程序代码
function dx = Derivada(h);
% 创建零值矩阵
dx=zeros(256,1);
% 通过函数公式来返回转换的数值
for i=3:254;
    dx(i)= (h(i-2) - 8*(h(i-1)) + 8*(h(i+1)) - h(i+2))/12;
end
%MaxiMini 子函数程序代码
function [Maxi, Mini] = MaxiMini(d);
j = 1; y = 1;
for i = 2 : 256;
    if (d(i - 1) < 0 && d(i) >= 0)
        Mini(j) = i;
        j = j + 1;
    elseif (d(i - 1) >= 0 && d(i) < 0)
        Maxi(y) = i;
        y = y + 1;
    end
end

function Vector = Repetidos(Arreglo, D);
N_Ind = length(Arreglo);
for i = 1 : N_Ind - 1;
    if Arreglo(i) > 0
        for j = i + 1 : N_Ind;
            if Arreglo(i) == Arreglo(j) || (abs(Arreglo(i)-Arreglo(j))<D)
                Arreglo_Nuevo(j) = 0;
            else
                Arreglo_Nuevo(j) = Arreglo(j);
            end
        end
    end
end
[i j Vector] = find(Arreglo_Nuevo);

function Umbral = Iterativo1(Imagen)
%=================================================================
% 检测图像文件的属性
if ~Esgray(Imagen)
    uitwait(msgbox('La imagen no se encuentra en escala de grises',
```

```
'Error','modal'))
    Umbral = 0;
else
%返回图像数据的直方图
Histograma = imhist(Imagen);
%返回直方图数据中的非零数值的下标
Grises = find(Histograma);
%返回最大下标和最小下标
    Maximo = max(Grises);
    Minimo = min(Grises);
    Umbral = Minimo + (Maximo - Minimo) / 2;
Umbralp = 0;
%使用循环进行图像转换
    while (abs(Umbral - Umbralp) > 1)
        Mayores = find(Imagen > Umbral);
        Menores = find(Imagen <= Umbral);
        m1 = mean(Imagen(Mayores));
        m2 = mean(Imagen(Menores));
        Umbralp = Umbral;
        Umbral = round((m1+m2)/2);
    end
    Umbral = Umbral/255;
end
%-------------------------------------------------------------------------
function y = Esgray(x)
%-------------------------------------------------------------------------
y = ndims(x)==2 && ~isempty(x);
if islogical(x)
  y = false;
elseif ~isa(x, 'uint8') && ~isa(x, 'uint16') && y
  % 选取图像的微小范围进行检测
  [m,n] = size(x);
  chunk = x(1:min(m,10),1:min(n,10));
  y = min(chunk(:))>=0 && max(chunk(:))<=1;
  %如果 chunk 是饱和度图像，则检查整个图像
  if y
    y = min(x(:))>=0 && max(x(:))<=1;
  end
end
```



在上面的程序代码中，主要定义了 Kapurl、Triangularl 和 Iterativol 三个图像转换函数分别对应的转换方法是 Kapur、Triangular 和 Iterative，关于这些方法的具体内容和操作信息，感兴趣的读者请自行查阅关于图形转换的内容。

step 4 查看 graythresh 函数的程序代码。在上面的程序代码中之所以没有另外定义 graythresh 函数，是因为该函数是 MATLAB 的内置函数，其具体代码如下：

```
function level = graythresh(I)
% 需要一个输入参数
%检测输入参数的数值
checknargin(1,1,nargin,mfilename);
checkinput(I,{'uint8','uint16' 'double'},{'nonsparse'},mfilename,'I',
1);
```

```
if ~isempty(I)
  % 将所有的数据数组转换为单列数据
  % 转换为 uint8 的数据格式，有利于图像转换的运算
  I = im2uint8(I(:));
  num_bins = 256;
  counts = imhist(I,num_bins);
   % 下面的参数名称和图像转换公式中的参数名称相同
  p = counts / sum(counts);
  omega = cumsum(p);
  mu = cumsum(p .* (1:num_bins)');
  mu_t = mu(end);
    previous_state = warning('off', 'MATLAB:divideByZero');
  sigma_b_squared = (mu_t * omega - mu).^2 ./ (omega .* (1 - omega));
  warning(previous_state);
  maxval = max(sigma_b_squared);
  if isfinite(maxval)
    idx = mean(find(sigma_b_squared == maxval));

    % 单位化转换单位的范围
    level = (idx - 1) / (num_bins - 1);
  else
    level = 0.0;
  end
else
  level = 0.0;
end
```

step 5 查看图形句柄检测函数。最后，除了设置对应的图像转换函数，还设置了检测图形句柄的函数 NoChecked，具体程序代码如下：

```
%=========================================================================
function NoChecked(handles)
%=========================================================================
% 将所有的菜单选项设置为没有选中
set(handles.graythresh,'checked','off')
set(handles.kapur,'checked','off')
set(handles.triangular,'checked','off')
set(handles.iterativo,'checked','off')
set(handles.DefinidoUsuario,'checked','off')
```

上面程序代码的功能是检测菜单选项对象是否被选中，然后根据检测的结果设置不同的菜单选项调用函数。



由于本小节的内容主要是介绍如何使用GUIDE创建菜单选项，同时设置菜单选项的属性，并且添加菜单选项的程序代码，因此关于上面程序的具体含义就不详细分析了，感兴趣的读者请直接查看关于M文件编程的章节。

11.5.6 添加滚动条的回调函数

延续上面小节的步骤。

step 1 添加"slider1"滚动条的回调函数。首先需要为滚动条添加回调函数，在 GUIDE 图形界面

中选中"slider1"对象，然后单击右键，在弹出的快捷菜单中选择"View Callbacks"⇨"Callback"命令，如图 11.59 所示。

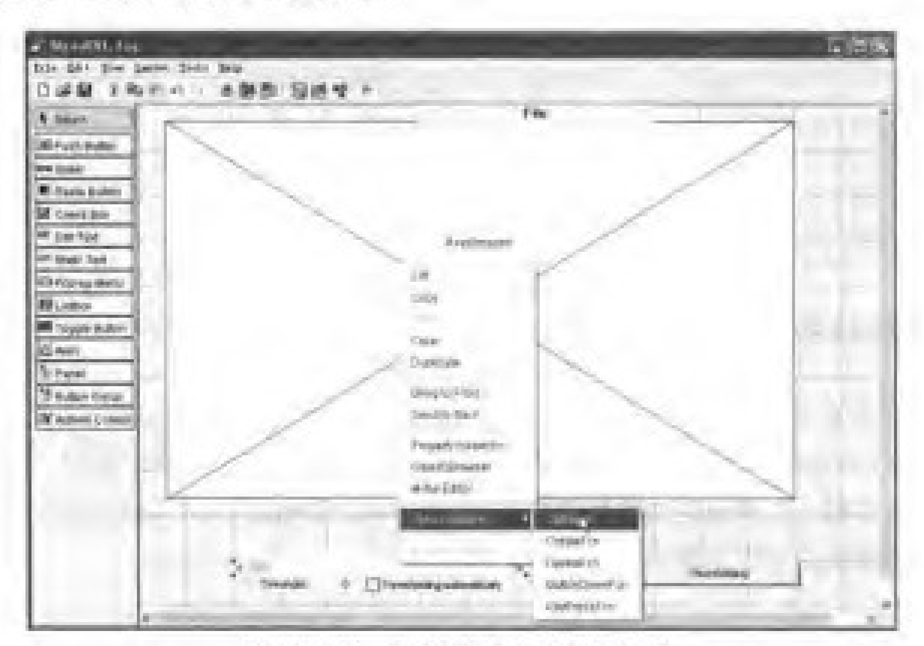


图 11.59 查看滚动条的回调函数

MATLAB 会自动跳到添加滚动条回调函数的位置，编写的程序代码如下：

```
% --- 当滚动条运动时将调用下面的程序代码
%----------------------------------------------------------------------
function slider1_Callback(hObject, eventdata, handles)
%----------------------------------------------------------------------
set(handles.Intensidad,'visible','on')
Umbral=get(hObject,'value');
% 调用 GraficarHistograma 函数
GraficarHistograma(Umbral,handles)
if get(handles.ActualAutomatico,'value')==1
% 调用 GraficarUmbral 函数
    GraficarUmbral(Umbral,handles);
end %if
NoChecked(handles);set(handles.DefinidoUsuario,'checked','on')
```

step 2 查看图形转换函数的程序代码。在上面的程序代码中，调用了两种图像转换函数，具体代码如下：

```
%======================================================================
function GraficarUmbral(Umbral,handles)
%======================================================================
global ImagenGris ImagenUmbral
% 将图像转换为二元制映像
imgUmbral=im2bw(ImagenGris,Umbral/255);
subplot(224);
% 显示图形
imshow(imgUmbral)
% 添加图形标题
title([ 'Image segmented in threshold = ',int2str(Umbral)])
ImagenUmbral=imgUmbral;
```

```
%======================================================================
function GraficarHistograma(Umbral,handles)
%======================================================================
global ImagenGris
persistent H
set(handles.Intensidad,'String',Umbral)
%绘制图形的直方图
subplot(223);imhist(ImagenGris);title('Histogram')
Escala=axis;
if ishandle(H)==1,delete(H),end%if
H=line([ Umbral Umbral] ,[ Escala(3:4)] ,'color','r');
```

 添加其他控件的回调函数

延续上面小节的步骤。

step 1 添加 “ActualAutomatico” 复选框的回调函数。在 GUIDE 图形界面中选中 “slider1” 对象，然后单击右键，在弹出的快捷菜单中选择 “View Callbacks” ⇒ “Callback” 命令，然后在 M 文件编辑器的对应位置编写代码如下：

```
% --- 当用户单击 ActualAutomatico 按钮后，触发该程序代码
%----------------------------------------------------------------------
function ActualAutomatico_Callback(hObject, eventdata, handles)
%----------------------------------------------------------------------
%设置 Umbralizar 对象句柄的属性
set(handles.Umbralizar,'enable','on')
set(handles.Umbralizar,'string','Thresholding')
set(handles.Umbralizar,'enable','off')
if get(hObject,'Value')==1
    set(handles.Umbralizar,'enable','off')
else
    set(handles.Umbralizar,'enable','on')
end %if
NoChecked(handles);set(handles.DefinidoUsuario,'checked','on')
```

step 2 添加 “Umbralizar” 按钮的回调函数。在 GUIDE 图形界面中选中 “Umbralizar” 对象，单击右键，在弹出的快捷菜单中选择 “View Callbacks” ⇒ “Callback” 命令，然后在 M 文件编辑器的对应位置编写代码如下：

```
% ---当用户单击 Umbralizar 按钮时触发下面的程序代码
%----------------------------------------------------------------------
function Umbralizar_Callback(hObject, eventdata, handles)
%----------------------------------------------------------------------
if isequal(get(handles.triangular,'checked'),'off')==1
Umbral=get(handles.slider1,'value');
%调用 GraficarUmbral 函数
GraficarUmbral(Umbral,handles);
%确定 DefinidoUsuario 对象的句柄被选中
    NoChecked(handles);set(handles.DefinidoUsuario,'checked','on')
else
%获取对象句柄的数值
    UmbralActual=get(handles.slider1,'value');
    Umbrales=get(handles.triangular,'userdata');
Pos=find(Umbrales==UmbralActual);
```

```
%返回错误信息
    if isempty(Pos),msgbox('Error de Ejecuci');return,end
if Pos==length(Umbrales),Pos=0;,end %if
%调用GraficarHistoUmbral函数
    GraficarHistoUmbral(Umbrales(Pos+1),handles,'')
end %if
```

step 3 查看图像转换函数的代码。在上面的程序代码中，调用了另外一种图像转换函数GraficarHistoUmbral，其详细的函数代码如下：

```
%==========================================================================
function GraficarHistoUmbral(Umbral,handles,Tipo)
%==========================================================================
%调用GraficarHistograma和GraficarUmbra函数
GraficarHistograma(Umbral,handles)
GraficarUmbral(Umbral,handles)
set(handles.slider1,'value',Umbral)
%设置对象的属性数值
switch Tipo
    case 'all' %GRAYTHRESH ITERATIVO KAPUR
        set(handles.ActualAutomatico,'value',0)
        set(handles.Umbralizar,'enable','on')
        set(handles.Umbralizar,'string','Thresholding')
        set(handles.Umbralizar,'enable','off')
    case 'trian' %TRIANGULAR
        set(handles.ActualAutomatico,'value',0)
        set(handles.Umbralizar,'enable','on')
        set(handles.Umbralizar,'string','next Threshold >>')
end %switch
```

编写主调函数

延续上面小节的步骤。

编写“CargarImagen”函数。其对应的程序代码如下：

```
%==========================================================================
function CargarImagen(handles)
%==========================================================================
global Archivo NameArchivo ImagenGris
%绘制原始图形
subplot(221);img=imread(Archivo);imshow(img);title('Original Image');
axis off
%绘制灰度图形
try
    if Esgray(img)==0
        imggris=rgb2gray(img);subplot(222);imshow(imggris);title('Gray
Scale Image')
    else
        imggris=img;subplot(222);imshow(imggris);title('Gray Scale
Image')
    end %if
catch
    errordlg('Error during image processing','Threshold GUI');error
('Error during image processing')
```

```
end %try
%绘制图像信息的直方图
subplot(223);imhist(imggris);title('Histogram')
ImagenGris=imggris;
Min=0;Max=255;
%设置控件的属性
set(handles.slider1,'visible','on')
set(handles.slider1,'Min',Min)
set(handles.slider1,'Max',Max)
set(handles.slider1,'value',Min)
set(handles.Umbralizar,'visible','on')
set(handles.ActualAutomatico,'visible','on')
set(handles.text2,'visible','on')
set(handles.Intensidad,'string',int2str(Min))
set(handles.Archivo,'visible','on')
set(handles.Archivo,'string',NameArchivo)
GraficarUmbral(Min,handles)
```

11.5.9 运行 GUI

延续上面小节的步骤。

step 1 选择图形文件。在 MATLAB 的命令窗口中输入"ThresGUI"，按"Enter"键，直接运行前面步骤中的 GUI 对象，如图 11.60 所示。



图 11.60　打开 GUI 对象

在上面的图形界面中，选择图形窗口中的"File"⇒"Open Image"命令，打开"Select File to Open"对话框，可以在该对话框中选择对应的图形文件，该程序支持的图形文件格式包括 jpg、tif、gif、bmp、png、hdf、pcx、xwd、ico、cur、ras、pbm、pgm、ppm 等。

step 2 运行 GUI 程序。当选中对应的文件后，单击对话框中的"打开"按钮，运行程序，如图 11.61 所示。

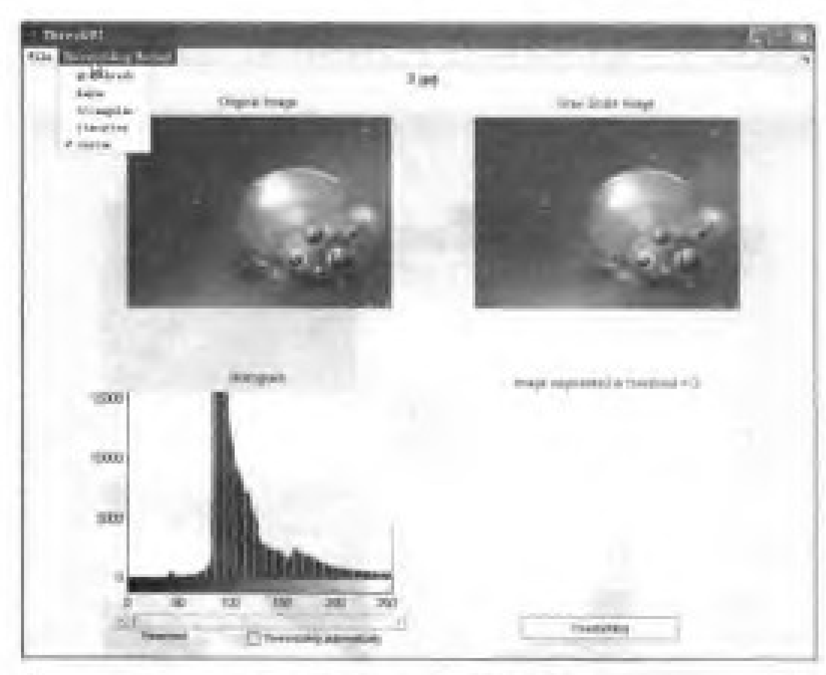


图 11.61 运行整个程序



从上面的图形界面中可以看出，当用户添加图形后，"Thresholding Method" 菜单选项被全部激活，用户可以选用对应的图像转换的方法。

step 3 选择图形转换类型。选择 "triangular" 菜单选项，然后在滚动条中选择图像转换参数，单击 "Thresholding" 按钮，得到的结果如图 11.62 所示。

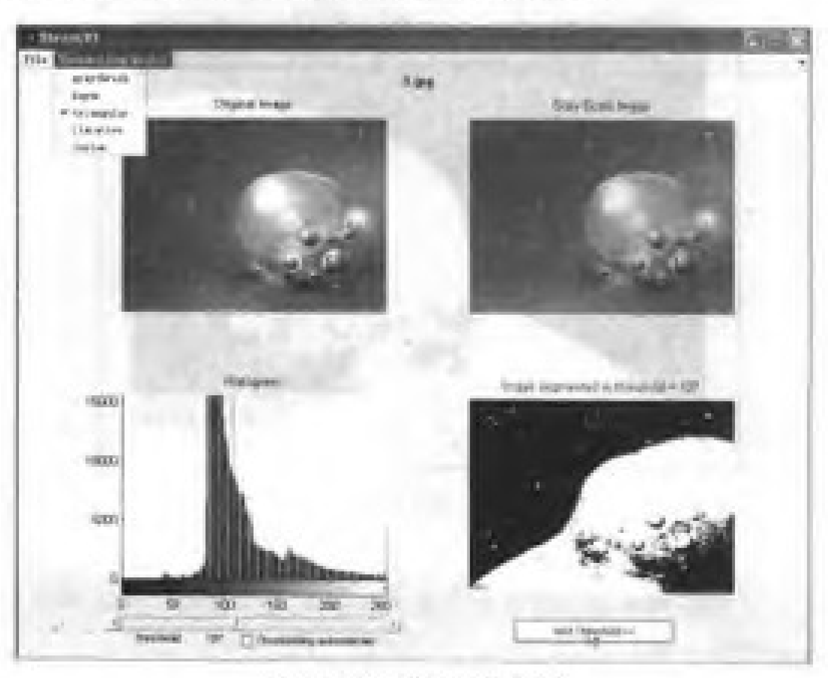


图 11.62 进行图像转换



单击 "Thresholding" 按钮后，图像转换的结果会出现在右下方的图形窗口中，同时，其按钮中的文字变为 "next Threshold>>"。连续单击该按钮，就可以得出不同图像转换的结果。

step 4 保存转换后的图形。当完成图形转换后，可以保存转换后的图形文件，如图 11.63 所示。

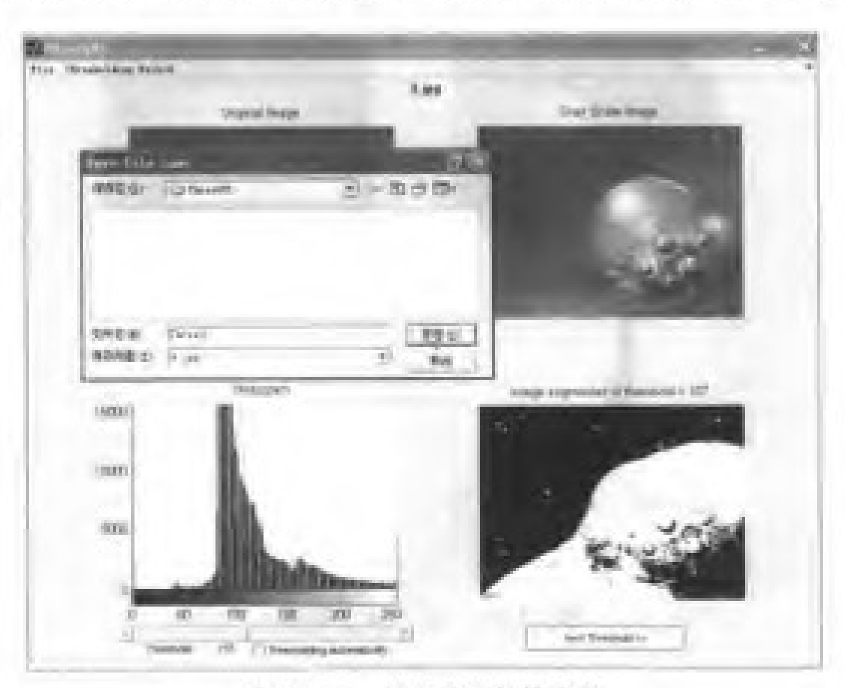


图 11.63　保存图形转换文件

选择 "File" ⇨ "Save Image" 命令，打开 "Save file name" 对话框，用户可以选择图形保存的路径和名称，设置保存属性后，单击 "保存" 按钮，就可以完成图形的保存工作。

step 5 查看保存图形。选择图形的保存路径，查看的图形结果如图 11.64 所示。



图 11.64　保存的图形文件



用户可以使用合适的图片浏览软件查看保存的图形文件，其保存的文件就是前面操作转换的图形文件。

step 6 退出 GUI 程序。选择图形窗口中的 "File" ⇨ "Close" 命令，打开对应的关闭对话框，单击对话框中的 "Yes" 按钮，就可以退出整个 GUI 程序，如图 11.65 所示。

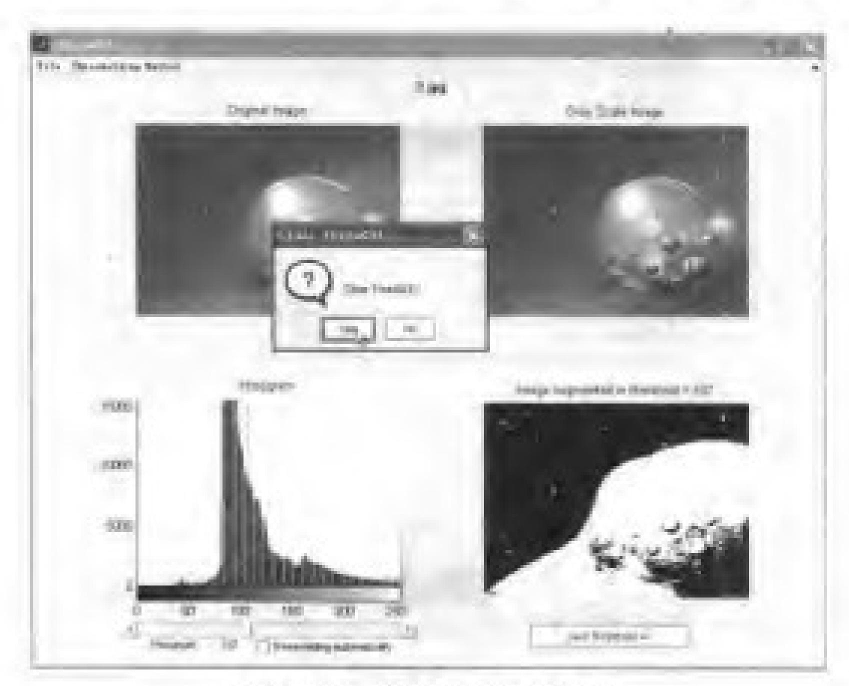


图 11.65 退出 GUI 应用程序



上面的简单实例演示了使用GUIDE创建MATLAB的菜单选项的步骤，首先通过"Menu Editor"对话框设置菜单的选项和外观属性，然后添加对应选项的回调函数，完成各种功能。

11.6 使用 M 文件创建自定义菜单

前面演示了使用GUIDE创建菜单选项的方法，在本小节中，将以一个另外的案例来演示如何使用M文件来创建自定义菜单。同样，该实例比较复杂，在本节中将分小节详细介绍。

11.6.1 演示 GUI 的功能

例 11.5 编写一个GUI对象，同时创建用户自定义的菜单，通过使用自定义菜单来完成文件操作，最后需要完成的功能是绘制文件数据图表。为了能够让读者大致了解该GUI对象的功能，下面简要演示GUI对象的操作过程。

step 1 选择处理文件。首先，"File"菜单可以完成相应的文件操作，如图11.66所示。



本例需要完成的GUI对象中，"File"菜单选项包括读入文本文件、读入Excel文件、保存文件和关闭GUI对象等所有文件操作选项，上面的文件操作是"读入Excel文件"菜单选项。

step 2 定义数据系列。选择"Options"菜单选项中的菜单选项，定义对应的X数据列、Y数据列、Z数据列、误差线数据列、数据标记等数据系列。在该步骤中，定义的是Z列数据，如图11.67所示。



图 11.66　读入数据文件



图 11.67　定义图表的数据系列

step 3 绘制数据系列。当用户已经读入数据文件，并定义图表的数据系列后，可以选择相应的菜单选项，绘制读入的数据系列，如图 11.68 所示。



图 11.68　选择绘图菜单

step 4 查看图形结果。当选择相应的绘图方式后，在图形窗口中就会显示绘图的结果，如图 11.69 所示。

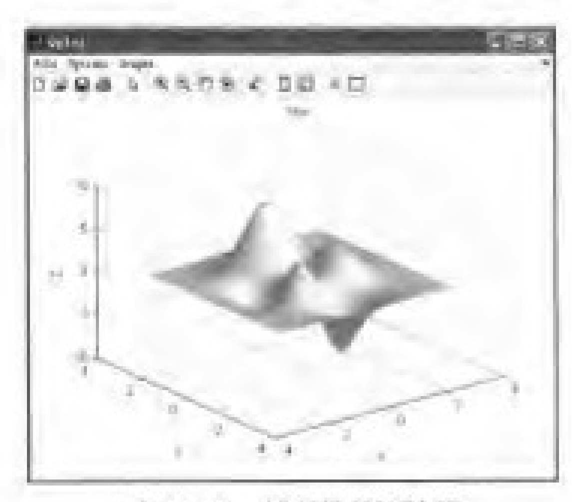


图 11.69 绘制的图形结果

从上面的演示过程中可以看出，在本GUI对象中所有的操作都是通过菜单选项完成的，该GUI对象中没有添加任何其他GUI的控件。同时，这些菜单选项都是用户自己创建设计的，不是MATLAB的标准菜单。



由于每一个菜单选项功能都需要编写对应的调用函数，为了更加方便用户理解整个系统的功能，在本实例中，将为每个调用函数编写一个M文件。最后，整个GUI对象将是一个包含多个M文件的集合文件。

11.6.2 添加“File”菜单的程序代码

从本小节开始，将详细介绍如何创建上面的GUI对象。

step 1 设计菜单选项的结构体系。由于在本实例中，所有的程序功能都是通过自定义菜单选项实现的，因此，有必要设计菜单选项的体系结构。

- **File菜单：** 该菜单中包括读取TXT文件和EXCEL文件，保存文件、退出系统等文件操作的所有菜单选项。因此，该File菜单包括Read Txt File、Read Excel File: All、Read Excel File: Selected Data、Save as和Quit菜单选项。
- **Options菜单：** 该菜单选项中包括添加数据的各种选项，用户可以从菜单选项中选择相应的选项，定义各种图表数据系列。
- **Graphs菜单：** 该菜单选项包括各种2D和3D绘图的选项，以及两种菜单选项下的各种绘图格式，这个菜单是绘图的主要菜单选项命令。

step 2 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function getdata (option)
global A opt leg label
%打开外部文件
[ fname,pname] = uigetfile('*.xls','Select File');
% 创建外部文件的完整路径字符串
dbfile= strcat(pname,fname);
% 如果没有打开文件，则跳出程序
```

```
if length(dbfile) == 0 return; end
%根据不同选项，读入文件信息
switch option
    case 1
        [A,leg]= readdata(dbfile);
    case 2
        [A,leg]= xlsread(dbfile);
    case 3
        [A,leg]= xlsread(dbfile,-1)
end

%创建元胞数组
if isempty(leg) leg= cellstr( num2str( flipud(rot90([1:size(A,2)])) )
); end
opt.xc= 1;
%根据读入数据系列的列数，决定参数数值
switch size(A,2)
    case 1
        opt.yc= 1;
        opt.ec= 1;
        opt.zc= 1;
    case 2
        opt.yc= 2;
        opt.ec= 2;
        opt.zc= 2;
    otherwise
        opt.yc= [2:size(A,2)-1];
        opt.ec= [fix(size(A,2)/2):size(A,2)];
        opt.zc= [3:size(A,2)];
end
return
```

将上面的程序代码保存为“getdata.m”，该程序代码将是File菜单选项中读取文件的菜单选项对应的程序代码。

step 3 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function savefigas
%保存文件
[fname,pname] = uiputfile('*.fig','Select File');
%创建保存文件的完整路径
dbfile= strcat(pname,fname);
%如果路径为空，则跳出程序代码
if length(dbfile) == 0 return; end
saveas(gcf,dbfile,'fig')
return
```

将上面的程序代码保存为“savefigas.m”，该程序代码将是File菜单选项中“Save as”的菜单选项对应的程序代码。



添加“Options”菜单的程序代码

延续上面小节的步骤。

step 1 单击MATLAB命令窗口工具栏中的🗋按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function getcols (col)
global A leg opt
% 如果读入的文件中没有数据，显示提示信息，并退出程序
if size(A) == 0    helpdlg('No data. You must read a file first',
'Error'); return, end
% 判断选择的数据列
switch col
% 如果选择的是x，则将数据读入变量opt的xc维中
    case 'x'
[ opt.xc,value]  = listdlg('PromptString','Choose X column',
'SelectionMode','single','ListString',leg);
% 如果选择的是y，则将数据读入变量opt的yc维中
    case 'y'
[ opt.yc,value]  = listdlg('PromptString','Choose Y column',
'SelectionMode','multiple','ListString',leg);
% 如果选择的是z，则将数据读入变量opt的zc维中
    case 'z'
[ opt.zc,value]  = listdlg('PromptString','Choose Z column',
'SelectionMode','multiple','ListString',leg);
% 如果选择的是e，则将数据读入变量opt的ec维中
    case 'e'
[ opt.ec,value]  = listdlg('PromptString','Choose Errors column',
'SelectionMode','multiple','ListString',leg);
end
return
```

将上面的程序代码保存为“getcols.m”，该程序代码将是Options菜单选项中定义数据系列的菜单选项对应的程序代码。

step 2 单击MATLAB命令窗口工具栏中的🗋按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function getlabel
global label
% 显示提示信息
prompt= { 'X label','Y label','Z label','Title'} ;
title= 'Axis Legends';
lines= 1;
resize= 'off';
tmp= inputdlg(prompt,title,lines,struct2cell(label));
fields= { 'x','y','z','t'} ;
if size(tmp,1) > 0 label= cell2struct(tmp,fields,1); end
return
```

将上面的程序代码保存为"getlabel.m"，该程序代码将是Options菜单选项中定义数据系列名称的菜单选项对应的程序代码。

step 1 单击MATLAB命令窗口工具栏中的🗋按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function window
```

```
global holdon
if (holdon) holdon= 0; else holdon= 1; end
if strcmp(get(gcbo, 'Checked'),'on')
    set(gcbo, 'Checked', 'off');
else
    set(gcbo, 'Checked', 'on');
end
return
```

将上面的程序代码保存为"window.m"，该程序代码将是Options菜单选项中"向图形中添加数据"的菜单选项对应的程序代码。

 添加"Graphs"菜单的程序代码

延续上面小节的步骤。

step 1 单击MATLAB命令窗口工具栏中的🗋按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function plot2d (type)
global A opt cds holdon leg label figmain
%当数组A 为空，或者数组A的第二维数值小于2
%显示对应的错误信息
if isempty(A) errordlg ('No data present. Sure you read a file yet?',
'Error'); return; end
if size(A,2) < 2 errordlg ('Seems to be only 1 column of data. Need at
least 2','Error'); return; end
%绘制图形
figure (figmain);
if (holdon)   hold on; else   hold off; end
%读入绘图的数据系列
X= A(:,opt.xc);
Y= A(:,opt.yc);
%选择图表类型，绘制对应的图表
switch type
    case 'xyscatter'
         plot(X,Y,'o');
    case 'xyline'
         plot(X,Y,'-o');
    case 'hist'
         hist(A(:,opt.yc(1)),10);
    case 'stem'
         stem(X,Y);
    case 'stairs'
         stairs(X,Y);
    case 'vbarg'
         bar(X,Y,'group');
    case 'vbars'
         bar(X,Y,'stack');
    case 'barerror'
%判断误差线的数据
         if size(Y,2) ~= size(A(:,opt.ec),2)
              msg= errordlg ('Y Data and Error Data have different number
of columns','Error');
              waitfor (msg);
```

```
                return;
            end
% 调用 barerror 函数，绘制误差线
            barerror(X,Y,A(:,opt.ec),0.8,'k');
        case 'hbars'
            barh(X,Y,'stack');
        case 'hbarg'
            barh(X,Y,'group');
        case 'rose'
            rose(X);
        case 'pie'
            pie(X);
        case 'polar'
            polar(X,A(:,opt.yc(1)));
        case 'compass'
            compass(X,A(:,opt.yc(1)));
        case 'error'
            if size(Y,2) ~= size(A(:,opt.ec),2)
                msg= errordlg ('Y Data and Error Data have different number
of columns','Error');
                waitfor (msg);
                return;
            end
            multX= [];
            for j= 0:opt.yc(2)-opt.yc(1) multX= [ multX,X] ; end
            errorbar(multX,Y,A(:,opt.ec));
end
title(label.t); xlabel(label.x); ylabel(label.y);
legend(leg(opt.yc));
return
```

将上面的程序代码保存为 "plot2d.m"，该程序代码将是Graphs菜单选项中 "2D" 的菜单选项对应的程序代码。



在上面的图形类别中，有一种图形类别比较特殊，这就是误差线类型。在上面的程序代码中，调用了函数 barerror 来绘制误差线，该函数需要用户单独定义。

step 2 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function barerror (X,Y,E,width,color)
% 判断绘图数组的大小
if mean([ size(X,1),size(Y,1),size(E,1)]) ~= length(X)    error ('Input
vectors are of different lengths'); return; end
if size(Y,2) ~= size(E,2) error ('Data and Error vectors have different
number of columns'); return; end
% 设定绘图颜色数组
colors= [ 'k';'w';'r';'g';'b';'c';'m';'y'];
hold on
ncol= size(Y,2);
off= [ fix(-ncol/2):fix(ncol/2)];
realwidth= min(diff(X))/(ncol);
if ~mod(ncol,2) off= [ off(1:ceil(length(off)/2)-1), off(1+ ceil(length
```

```
(off)/2):length(off))]·; end
for h= 1:ncol
    Xtmp= X(:,1)+ off(h)*(realwidth/2)- sign(off(h))*(~mod(ncol,2)
*realwidth/4);
%绘制直方图
    bar(Xtmp,Y(:,h),width/(2*ncol),colors(mod(h,1+length(colors))));
%绘制误差线
    errorbar(Xtmp,Y(:,h),E(:,h),'LineStyle','none','Color',color);
end
hold off
return
```

将上面的程序代码保存为"barerror.m"，该程序代码将是Graphs菜单选项中"2D"的菜单选项下"barerror"菜单选项对应的程序代码。

step 3 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function plot3d (type)
global A opt holdon label figmain
%判断数据系列的维度
if isempty(A) errordlg ('No data present. Sure you read a file yet?',
'Error'); return; end
if size(A,2) < 3 errordlg ([ 'Seems to be only ',num2str(size(A,2)),'
columns of data. Need at least 3'],'Error'); return; end
%绘制图形
figure (figmain);
if (holdon)   hold on; else   hold off; end
%读入数据系列
X= A(:,opt.xc(1));
Y= A(:,opt.yc(1));
M= A(:,opt.zc);
%当数据系列的维度不匹配时，显示错误信息
if size(M,2) > 1 & size(M,2) < length(X)
    warn= msgbox('Number of Z columns > 1 but < number of rows in X.
Only the 1st Z column displayed','Warning');
    waitfor(warn);
    M= A(:,opt.zc(1));
end

if size(M,2) == 1
   [XI,YI]= meshgrid(min(X):range(X)/(length(X)-1):max(X),min(Y):range
(Y)/(length(Y)-1):max(Y));
    Mitp= griddata(X,Y,M,XI,YI);
end
%选择绘制图形的图表类型
switch type
    case 'waterfall'
         if size(M,2) ~= 1 waterfall(X,Y,M); end
         if size(M,2) == 1 waterfall(XI,YI,Mitp);
         end
    case 'ribbon'
         ribbon(X,M);
    case 'grid'
         if size(M,2) ~= 1 mesh(X,Y,M); end
```

```
            if size(M,2) == 1 mesh(XI,YI,Mitp);
            end
        case 'bar3'
            if size(M,2) ~= 1 bar3(X,M,'detached'); end
            if size(M,2) == 1 bar3(X,Mitp,'detached'); end
        case 'plot3'
            plot3(X,Y,M,'o');
        case 'stem3'
            stem3(X,Y,M(:,1));
        case 'surface'
            if size(M,2) ~= 1 surf(X,Y,M); end
            if size(M,2) == 1 surf(XI,YI,Mitp);
            end
        case 'smooth'
            if size(M,2) ~= 1 surfl(X,Y,M); end
            if size(M,2) == 1 surfl(XI,YI,Mitp); end
            shading interp;
            colormap(pink);
        case 'contour'
            if size(M,2) ~= 1 contour(X,Y,M,10); end
            if size(M,2) == 1 contour(XI,YI,Mitp,10); end
end
%添加图表的标题和坐标轴名称
title(label.t); xlabel(label.x); ylabel(label.y); zlabel(label.z);
grid on;
return
```

将上面的程序代码保存为“plot3d.m”，该程序代码将是Graphs菜单选项中“3D”的菜单选项对应的程序代码。



添加主调函数

延续上面小节的步骤。

step 1 单击MATLAB命令窗口工具栏中的□按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
function Qplot()
clear all
warning off
global A cds opt holdon leg label figmain
% opt.xc= 1; opt.yc= 2; opt.ec= 3; opt.zc= 3
% 创建opt和label结构体
opt= struct('xc',1,'yc',2,'ec',3,'zc',3);
label= struct('x','X','y','Y','z','Z','t','Title');
holdon= 0;
fullm= 0;
% leg= struct('tex','');
fp = get(0,'defaultfigureposition');
% fp = [ fp(1)-150 fp(2)+fp(4)-1 150 1];
%创建原始的空白图形窗口
figmain= figure ('menubar','None','Toolbar','figure','Name','Qplot',
'Resize','On','NumberTitle','off','Color','white','Position',fp);
% 创建 "File" 菜单选项
mfile= uimenu('Label','File');
```

```
    uimenu(mfile,'Label','Read Text File','Callback','getdata(1)',
'Accelerator','R');
    uimenu(mfile,'Label','Read Excel File: All','Callback','getdata(2)
','Accelerator','E');
    uimenu(mfile,'Label','Read Excel File: Selected Data','Callback',
'getdata(3)','Accelerator','I');
    uimenu(mfile,'Label','Save as...','Callback','savefigas',
'Accelerator','S');
    uimenu(mfile,'Label','Quit','Callback','exit','Separator','on',
'Accelerator','Q');
%创建 "Options" 菜单选项
mopt= uimenu('Label','Options');
    uimenu(mopt,'Label','X column  (def. 1)','Callback','getcols(''x'')
','Accelerator','X')
    uimenu(mopt,'Label','Y columns','Callback','getcols(''y'')',
'Accelerator','Y')
    uimenu(mopt,'Label','Z column','Callback','getcols(''z'')',
'Accelerator','Z')
    uimenu(mopt,'Label','Error column','Callback','getcols(''e'')')
    uimenu(mopt,'Label','Axis Labels','Callback','getlabel',
'Accelerator','L')
    uimenu(mopt,'Label','Add graph to plot','Callback','window',
'Accelerator','A')
%创建 "Graphs" 菜单选项
graph= uimenu('Label','Graphs');
%创建 "2D" 子菜单选项
    m2d= uimenu(graph,'Label','2D');
          uimenu(m2d,'Label','XY Scatter','Callback','plot2d
(''xyscatter'')');
          uimenu(m2d,'Label','XY Line','Callback','plot2d(''xyline'')');
          uimenu(m2d,'Label','XY Line with error bar','Callback','plot2d
(''error'')');
          uimenu(m2d,'Label','Horizontal Bar (grouped)','Callback',
'plot2d(''hbarg'')');
          uimenu(m2d,'Label','Horizontal Bar (stacked)','Callback',
'plot2d(''hbars'')');
          uimenu(m2d,'Label','Vertical Bar (grouped)','Callback','plot2d
(''vbarg'')');
          uimenu(m2d,'Label','Vertical Bar (stacked)','Callback','plot2d
(''vbars'')');
          uimenu(m2d,'Label','Vertical Bar with error bars','Callback',
'plot2d(''barerror'')');
          uimenu(m2d,'Label','Histogram','Callback','plot2d(''hist'')');
          uimenu(m2d,'Label','Stem','Callback','plot2d(''stem'')');
          uimenu(m2d,'Label','Stairs','Callback','plot2d(''stairs'')');
          uimenu(m2d,'Label','Rose','Callback','plot2d(''rose'')');
          uimenu(m2d,'Label','Polar','Callback','plot2d(''polar'')');
          uimenu(m2d,'Label','Compass','Callback','plot2d(''compass'')');
          uimenu(m2d,'Label','Pie','Callback','plot2d(''pie'')');
%创建 "3D" 子菜单选项
m3d= uimenu(graph,'Label','3D');
          uimenu(m3d,'Label','Scatter 3D','Callback','plot3d(''plot3'')
');
          uimenu(m3d,'Label','Stem 3D','Callback','plot3d(''stem3'')');
          uimenu(m3d,'Label','Bar 3D','Callback','plot3d(''bar3'')');
          uimenu(m3d,'Label','Waterfall','Callback','plot3d
```

```
(''waterfall'')');
        uimenu(m3d,'Label','Ribbon','Callback','plot3d(''ribbon'')');
        uimenu(m3d,'Label','Grid','Callback','plot3d(''grid'')');
        uimenu(m3d,'Label','Surface','Callback','plot3d(''surface'')');
        uimenu(m3d,'Label','Smooth Surface','Callback','plot3d
(''smooth'')');
        uimenu(m3d,'Label','Contour','Callback','plot3d(''contour'')');
return
```

将上面的程序代码保存为“Qplot.m”，该程序代码将是该GUI对象的主程序代码。

step 2 分析程序代码。在上面的程序代码中，反复使用uimenu命令来创建各种菜单选项。该命令是在MATLAB中创建菜单对象的主要命令，下面选取其中的部分语句来分析该命令的用法。

```
mopt= uimenu('Label','Options');
...................................
uimenu(mopt,'Label','Y columns','Callback','getcols(''y'')',
'Accelerator','Y')
```

在上面的命令行中，首先使用unimenu命令创建一个标题为“Options”的菜单选项，同时将该菜单选项的句柄赋值给变量mopt；然后使用unimenu命令创建“Y columns”菜单选项，该菜单选项对应的回调函数是“getcols("y")”，同时，该菜单选项的快捷键为“Ctrl + Y”，该快捷键标记会出现在菜单名称后面。



在MATLAB中，菜单选项的快捷键只有在菜单选项不可见时才能有效。

11.6.6 演示GUI对象

延续上面小节的步骤。

step 1 读入Excel文件。上面的步骤已经完成所有的菜单选项的设置，从这个步骤开始，开始演示程序代码。首先读入预先准备好的“Qplot”Excel文件，如图11.70所示。

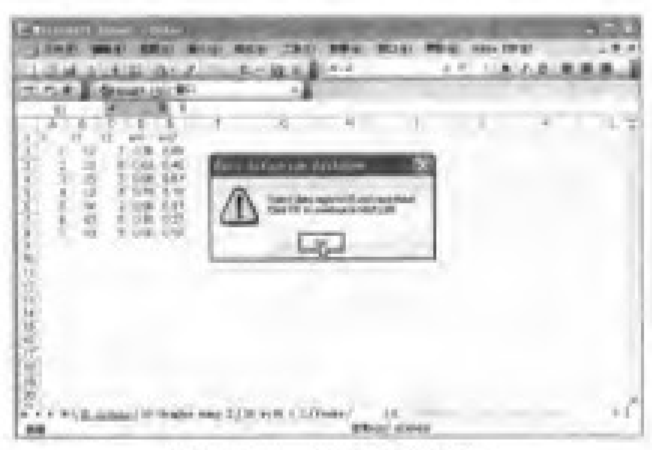


图11.70　读入数据文件

step 2 定义数据系列。当读入数据文件后，可以定义相应的数据系列，在本例中除了需要添加X和Y数据系列之外，还需要添加误差线数据系列，如图11.71所示。



图 11.71　添加误差线数据系列

step 3 添加图形标题。在添加所有的数据系列后，用户需要为图形添加图形的标题，选择"Options"⇨"Axis Labels"命令，打开"Axis Legends"对话框，在其中设置图形的标题，如图 11.72 所示。



图 11.72　添加图形的标题

step 4 选择绘图类型。当设置图形的标题后，需要选择对应的绘图类型，在本例中需要绘制的是误差线的直线，因此选择"Graphs"⇨"2D"⇨"XY Line with error bar"命令，如图 11.73 所示。



图 11.73　选择绘制的图形类型

step 5 查看图形结果。当用户选择相应的菜单选项后，MATLAB 就会绘制出对应的图形，如图 11.74 所示。

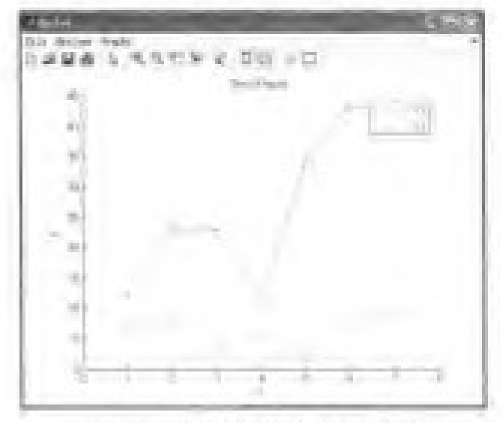


图 11.74 绘制的图形结果

step 6 重新选择绘图类型。对于同一个数据文件，用户可以选择不同的图形类型，来绘制该数据变化趋势。例如，在本实例中可以选择 "Graphs" ⇨ "2D" ⇨ "Vertical Bar with error bars" 命令，查看重新绘制的图形结果，如图 11.75 所示。

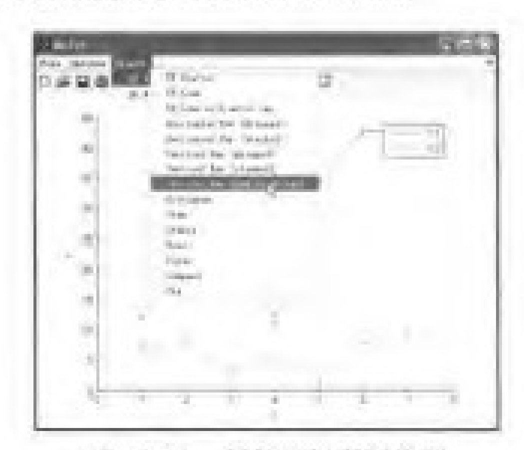


图 11.75 重新选择图形类型

step 7 重新查看图形结果。当用户选择对应的图形类型后，MATLAB 就会绘制出对应的图形，如图 11.76 所示。

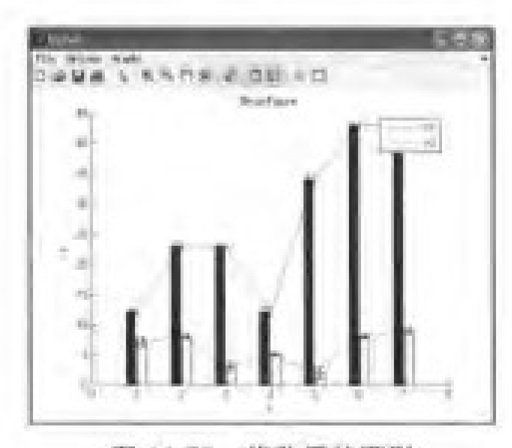


图 11.76 修改后的图形



用户还可以演示菜单选项的其他功能，限于篇幅，这里就不详细介绍这些内容了。

11.7 创建现场菜单

前面详细介绍了如何创建菜单栏的菜单选项，本节将介绍如何创建现场菜单。现场菜单就是当用户用右键选择对象后，弹出的快捷菜单选项。因此，可以认为现场菜单都和某个图形对象相联系，并通过鼠标右键来激活。

创建现场菜单的一般步骤如下：

- 利用命令 uicontextmenu 创建现场菜单对象；
- 利用命令 uimenu 来设置该现场菜单对象的具体属性；
- 利用命令 set 将现场菜单和图形对象联系起来。

本节将利用一个比较简单的实例来介绍如何在 MATLAB 中创建现场菜单。现场菜单也是属于 MATLAB 的图形对象之一，因此可以使用 GUIDE 来创建现场菜单，也可以直接使用 M 文件来创建现场菜单。

在本实例中，将直接使用 M 文件来创建现场菜单，下面分小节详细介绍。

11.7.1 编写 GUI 的程序代码

例 11.6 创建一个图形对象，然后创建一个与之联系的现场菜单，该菜单可以控制关于图形的各种属性。为了让读者直观地了解该现场菜单的属性，在MATLAB的命令窗口中输入下面的程序代码：

```
>> imagesc(peaks);
>> axis image;
>> imagemenu
```

上面的程序代码就可以在图形对象中添加现场菜单，如图 11.77 所示。

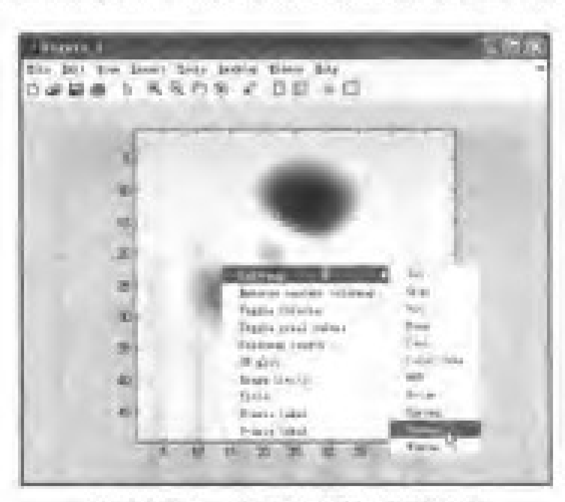


图 11.77 演示完成的现场菜单

在上面的现场菜单中，可以控制图形的颜色系统、转换图像方向、显示图形的颜色条、绘制三维图形、设置图形的标题和坐标轴名称等，下面分步骤详细介绍创建过程。

step 1 单击MATLAB命令窗口工具栏中的按钮，打开M文件编辑器。在M文件编辑器中输入下面的程序代码：

```
% Menu 的回调函数
function togglecolorbar(obj, eventdata)
%确定是否在图形界面中包含颜色条对象
phch = get(findall(gcf,'type','image','tag','TMW_COLORBAR'),{'parent'});
for i=1:length(phch)
    phud = get(phch{i},'userdata');
    if isfield(phud,'PlotHandle')
        if isequal(gca, phud.PlotHandle)
            delete(phch{i})
            axis image
            return
        end
    end
end

% 如果没有，则创建颜色条
colorbar
axis image
```

上面的程序代码的主要功能是在图形对象中显示颜色条，该程序代码对应的菜单选项是“Togglecolorbar”。在上面的程序代码中，首先判断图形中是否有图形颜色条，如果没有颜色条，则向其中添加颜色条对象。

step 2 在M文件编辑器中添加下面的程序代码：

```
% Menu 回调函数
function colormaplength(obj, eventdata)
%获取当前图形的色图
cmap = colormap;
%获取色图的像素单位
oldlength = length(cmap);
clength = cellstr(num2str(oldlength));
%提示用户输入新的像素单位
new = inputdlg({'Enter new colormap length:'}, ...
    'New colormap length', 1, clength);
newlength = str2double(new{1});
oldsteps = linspace(0, 1, oldlength);
newsteps = linspace(0, 1, newlength);
newmap = zeros(newlength, 3);

for i=1:3
    % 在RGB图形体系下对图像的像素进行插值
    newmap(:,i) = min(max(interp1(oldsteps, cmap(:,i), newsteps)', 0),
1);
end
%采用新的像素单位
colormap(newmap);
%如果在图形界面中包含了颜色条，对其更新
phch = get(findall(gcf,'type','image','tag','TMW_COLORBAR'),{'parent'});
for i=1:length(phch)
    phud = get(phch{i},'userdata');
```

```
    if isfield(phud,'PlotHandle')
        if isequal(gca, phud.PlotHandle)
            colorbar
        end
    end
end
```

上面代码的功能是重新设置图形像素的单位，该程序代码对应的菜单选项是“Colormap length”。当用户选择该菜单选项后，MATLAB就会显示出对话框，提示用户输入图形像素的新单位，确定新的图形像素后，将使用该单位绘制图形。

step 3 在M文件编辑器中添加下面的程序代码：

```
% Menu 回调函数
function call3d(obj, eventdata)
%返回当前图形的坐标轴对象
ax = gca;
temp = double(get(gco, 'CData'));
%创建新的图形窗口
newfig = figure;
newax = axes;
if isempty(get(get(ax, 'Parent'), 'Name'))
    set(newfig, 'Name','3D view');
else
    set(newfig, 'Name', [ get(get(ax, 'Parent'), 'Name') ', 3D view'] );
end
%绘制曲面图
s = surf(temp, 'LineStyle', 'none');
hl = camlight;
%添加坐标轴名称
xlabel('X distance [ pixels] ');
ylabel('Y distance [ pixels] ');
axis('tight')
```

上面代码的功能是绘制当前图形数据的三维图形，该程序代码对应的菜单选项是“3D plot”。当用户选择该菜单选项后，MATLAB就会在新的图形窗口中绘制新图形，而且默认情况下的图形类型是surf曲面图。

step 3 在M文件编辑器中添加下面的程序代码：

```
% Menu 回调函数
function imagelimits(obj, eventdata)
%获取colormap像素数值范围
lims = get(gca, 'CLim');
oldlower = num2str(lims(1));
oldupper = num2str(lims(2));
%显示对话框，提示用户输入新的数值范围
new = inputdlg({ 'Enter new lower limit:', 'Enter new upper limit:'}, ...
    'New image limits', 1, { oldlower, oldupper} );
if ~isnan(str2double(new{ 1} )) & ~isnan(str2double(new{ 2} ))
    set(gca, 'CLim', [ str2double(new{ 1} ) str2double(new{ 2} )] );
end

% 如果图形窗口中有颜色条，更新该颜色条
phch = get(findall(gcf,'type','image','tag','TMW_COLORBAR'),{ 'parent'} );
```

```
for i=1:length(phch)
    phud = get(phch{ i} ,'userdata');
    if isfield(phud,'PlotHandle')
        if isequal(gca, phud.PlotHandle)
            colorbar
        end
    end
end
```

上面代码的功能是设置图形中colormap数值范围，对应的菜单选项为“Image limits”。当用户选中该选项后，MATLAB会显示相应的对话框，用户可以在对话框中设置上限和下限，然后系统会按照新的数值范围绘制图形。

step 5 在M文件编辑器中添加下面的程序代码：

```
% Menu 回调函数
function titlecallback(obj, eventdata)
%获取原始图形窗口的标题属性
old = get(gca, 'title');
oldstring = get(old, 'string');
if ischar(oldstring)
    oldstring = cellstr(oldstring);
end
%提示用户输入新的标题
new = inputdlg('Enter new title:', 'New image title', 1, oldstring);
%设置新的标题
set(old, 'string', new);

% Menu 回调函数
function xaxiscallback(obj, eventdata)
%获取原始图形窗口的x轴名称
old = get(gca, 'xlabel');
oldstring = get(old, 'string');
if ischar(oldstring)
    oldstring = cellstr(oldstring);
end
%输入新的X轴名称
new = inputdlg('Enter new X-axis label:', 'New image X-axis label', 1,
oldstring);
%设置新的x轴名称
set(old, 'string', new);

% Menu callback
function yaxiscallback(obj, eventdata)
%获取原始图形窗口的y轴名称
old = get(gca, 'ylabel');
oldstring = get(old, 'string');
if ischar(oldstring)
    oldstring = cellstr(oldstring);
end
%输入新的y轴名称
new = inputdlg('Enter new Y-axis label:', 'New image Y-axis label', 1,
oldstring);
%设置新的y轴名称
set(old, 'string', new);
```

上面代码的功能是设置图形的标题和X、Y坐标轴的名称，对应的函数分别为titlecallback、xaxiscallback和yaxiscallback，对应的菜单选项为“Title”、“X-axis Label”和“Y-axis Label”。当用户选用该菜单选项后，会弹出相应的对话框，用户可以在对话框中设置图形的标题、X坐标轴和Y坐标轴的名称。

step 6 在M文件编辑器中添加下面的程序代码：

```
function imagemenu(handle)
%  示例
%   imagesc(peaks)
%   axis image
%   imagemenu
if nargin == 0
    % Use all images in current figure as default
    handle = gcf;
end

handle = findobj(handle, 'type', 'image');

% 定义现场菜单
cmenu = uicontextmenu;
% 定义现场菜单子选项
colormapmenu = uimenu(cmenu, 'Label', 'Colormap');
uimenu(cmenu, 'Label', 'Reverse current colormap', 'Callback', 'colormap
(flipud(colormap))');
uimenu(cmenu, 'Label', 'Toggle colorbar', 'Callback', @togglecolorbar);
if exist('pixval.m')
        uimenu(cmenu, 'Label', 'Toggle pixel values', 'Callback',
'pixval');
end
uimenu(cmenu, 'Label', 'Colormap length...', 'Callback', @colormaplength);
uimenu(cmenu, 'Label', '3D plot...', 'Callback', @call3d);
uimenu(cmenu, 'Label', 'Image limits...', 'Callback', @imagelimits);
uimenu(cmenu, 'Label', 'Title...', 'Callback', @titlecallback);
uimenu(cmenu, 'Label', 'X-axis label...', 'Callback', @xaxiscallback);
uimenu(cmenu, 'Label', 'Y-axis label...', 'Callback', @yaxiscallback);

% 定义"colormapmenu"菜单的子选项
uimenu(colormapmenu, 'Label', 'Jet', 'Callback', 'colormap(jet)');
uimenu(colormapmenu, 'Label', 'Gray', 'Callback', 'colormap(gray)');
uimenu(colormapmenu, 'Label', 'Hot', 'Callback', 'colormap(hot)');
uimenu(colormapmenu, 'Label', 'Bone', 'Callback', 'colormap(bone)');
uimenu(colormapmenu, 'Label', 'Cool', 'Callback', 'colormap(cool)');
uimenu(colormapmenu, 'Label', 'Color cube', 'Callback', 'colormap
(colorcube)');
uimenu(colormapmenu, 'Label', 'HSV', 'Callback', 'colormap(hsv)');
uimenu(colormapmenu, 'Label', 'Prism', 'Callback', 'colormap(prism)');
uimenu(colormapmenu, 'Label', 'Spring', 'Callback', 'colormap(spring)');
uimenu(colormapmenu, 'Label', 'Summer', 'Callback', 'colormap(summer)');
uimenu(colormapmenu, 'Label', 'Winter', 'Callback', 'colormap(winter)');
% 将菜单对象添加到图形句柄中
set(handle, 'uicontextmenu', cmenu);
```

将全部的程序代码保存为“imagemenu.m”文件。在程序代码中，首先使用uicontextmenu命令

创建一个现场菜单选项，然后使用 uimenu 命令设置现场菜单选项的属性，最后使用 set 命令将该现场菜单选项和图形句柄联系起来。

11.7.2 演示 GUI 对象

延续上面小节的步骤。

step 1 演示完成的程序代码。在命令窗口中输入下面的代码：

```
>> imagesc(sphere(50))
>>axis image
>>imagemenu
```



在上面的程序代码中，首先使用 imagesc 命令来显示一个图形对象，然后使用 axis 设置图形对象的坐标轴属性，最后在上面的图形中添加现场菜单。

step 2 修改图形的颜色模式。选中程序代码绘制的图形对象，单击鼠标右键，在弹出的快捷菜单中选择 "Colormap" ⇨ "Spring" 命令，修改图形的颜色模式，如图 11.78 所示。



图 11.78 修改图形的颜色模式

step 3 查看图形结果。选择上面的菜单选项后，得出修改后的图形如图 11.79 所示。

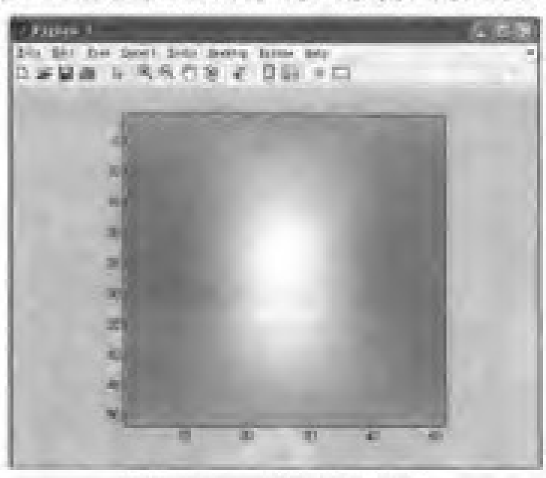


图 11.79 修改后的图形

step 4 向图形界面添加“颜色条”。单击鼠标右键，在弹出的快捷菜单中选择“Toggle colorbar”命令，如图 11.80 所示。

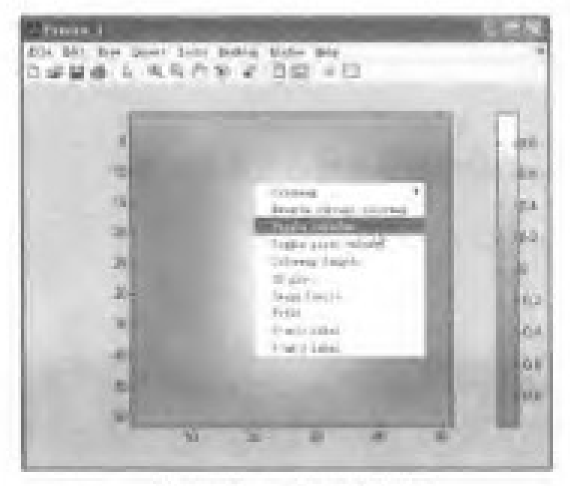


图 11.80　添加颜色条

step 5 添加显示图形像素的对话框。单击鼠标右键，在弹出的快捷菜单中选择“Toggle pixel values”命令，如图 11.81 所示。

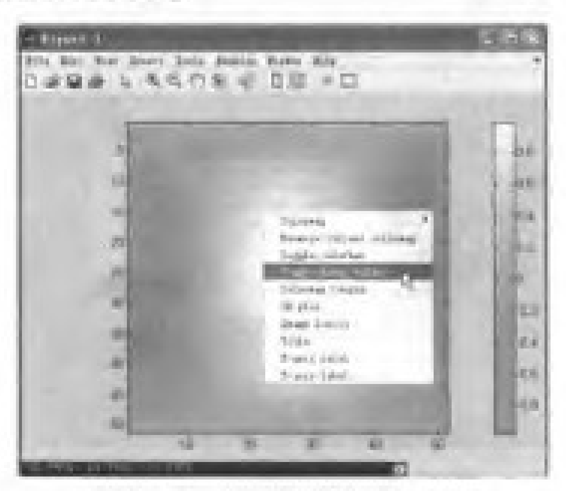


图 11.81　显示图形的像素数值

step 6 设置新的图形像素数值。单击鼠标右键，在弹出的快捷菜单中选择“Colormap length”命令，弹出“New colormap length”对话框，在其中可以设置新的图形像素数值，如图 11.82 所示。

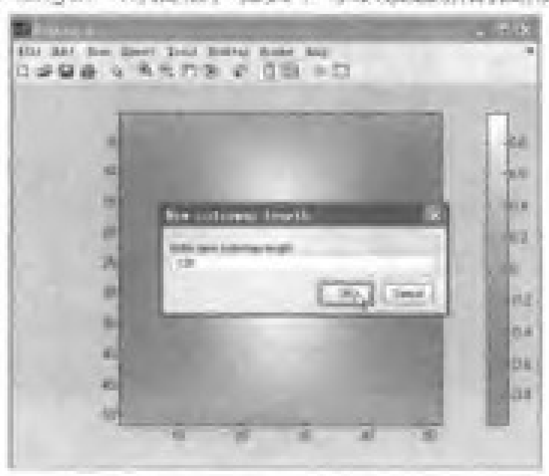


图 11.82　设置新的图形像素数值

step 7 查看三维图形结果。单击鼠标右键，在弹出的快捷菜单中选择"3D plot"命令，MATLAB就会在新的图形界面中显示三维图形，如图11.83所示。

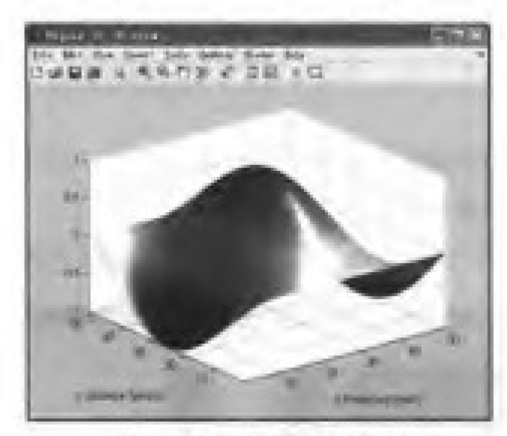


图 11.83　显示三维图形

step 8 重新设置图形界面的像素上下限。选择图形对象，然后单击鼠标右键，在弹出的快捷菜单中选择"Image limits"命令，弹出"New image limits"对话框，在其中设置图形对象像素的上下限，如图11.84所示。

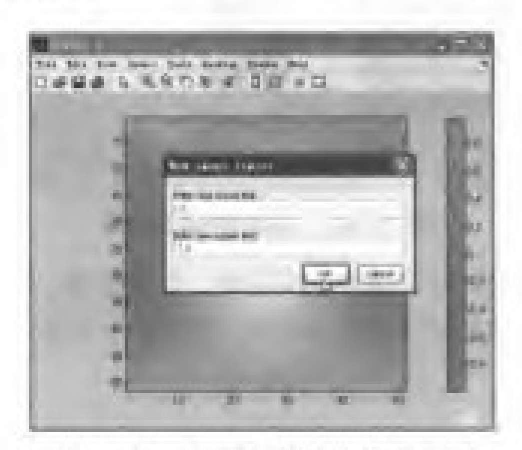


图 11.84　设置新的图形像素范围

step 9 添加坐标轴的名称。最后，用户可以使用对应的菜单选项，添加图形的标题和X、Y坐标轴的名称，得到的最后结果如图11.85所示。

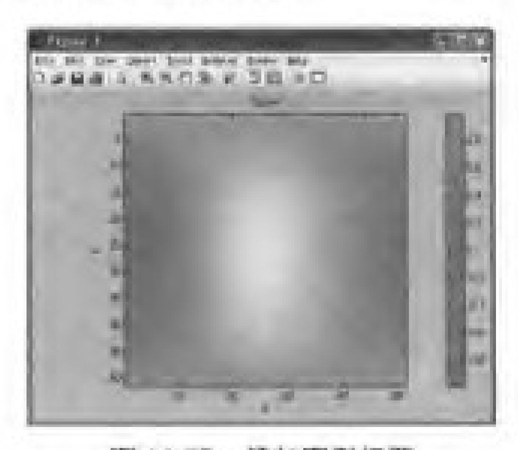


图 11.85　添加图形标题

11.8 创建GUI对象的用户控件

前面已经多次接触过GUI对象的用户控件。在GUI中，用户控件是除了用户菜单之外，实现用户和计算机交互的重要途径。在本章的第一节中，用户已经接触过GUI的控件面板，该面板中列举了所有可用的用户控件类型。

本节将以一个比较简单的例子来介绍在MATLAB中创建GUI用户控件的方法，在介绍具体方法的同时将介绍关于控件属性的各种基本内容。

例 11.7 编写一个关于图形三维显示的GUI，完成后的GUI如图11.86所示。

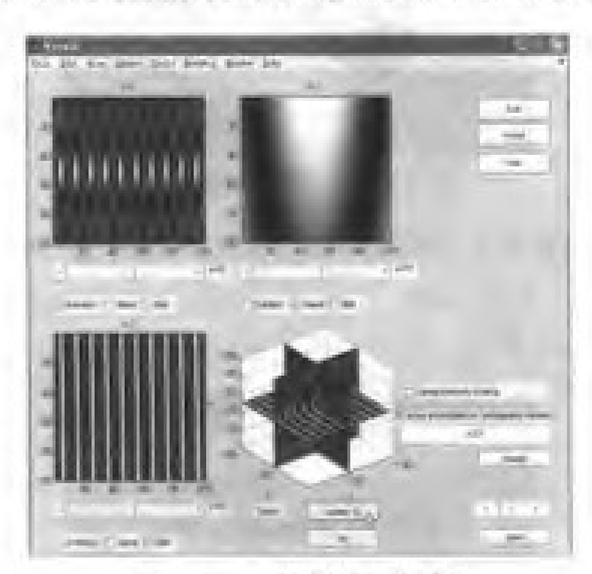


图 11.86 完成的GUI对象

在上面的GUI中，包括了MATLAB中多种用户控件，包括按钮、滚动条、复选框、编辑框、坐标轴等。由于这些控件类型都是MATLAB中的常用控件，下面分小节来介绍各种控件的创建方法和属性设置方法。

11.8.1 添加控件组件

在本小节中，将分步骤详细介绍如何添加各种控件组件。

step 1 打开GUIDE，然后向GUIDE中添加坐标轴控件。在控件面板中选择"Axes"控件，然后将其添加到GUIDE中，如图11.87所示。



图 11.87 添加坐标轴控件

step 2 添加滚动条（Slider）控件。在控件面板中选择"Slider"控件，然后将其添加到GUIDE中，如图11.88所示。



图11.88 添加滚动条控件

在MATLAB中，滚动条控件的主要功能是使用户能够通过滚动条来改变指定范围内的数值输入，滚动条的位置代表用户输入的数值。在本实例中，该滚动条的功能是修改图形截面在三维坐标系中的位置坐标。

step 3 分析滚动条控件的功能。对于滚动条控件，其属性"Max"和"Min"的数值决定了滚动条数值范围的上下限，另外，属性"SliderStep"是一个二元数组，第一个元素决定两端箭头操作滚动条时的步长，第二个元素则决定游标操作滚动条时的步长。关于滚动条控件，还有一个重要的属性"Value"，其对应的属性和游标位置相对应。



关于上面介绍的内容，在本小节中将不会进行具体操作。关于该控件属性的详细设置情况，将在后面章节中进行介绍。

step 4 添加"静态文本"控件。在控件面板中选择"Static Text"控件，将其添加到GUIDE中，然后打开该控件的属性列表对话框，将其"String"属性设置为"_"，如图11.89所示。



图11.89 添加"静态文本"控件

该静态文本的作用在于显示滚动条的数值增长方向，因此本控件对象对应的字符串文字表示：当用户向右移动该静态文本时，其对应的数值是减少的。

step 5 添加"复选框"控件。在控件面板中选择"Check Box"控件，将其添加到GUIDE中，然后打开该控件的属性列表对话框，将其"String"属性设置为"transpose"，将其背景颜色设置为白色，如图11.90所示。



图 11.90　添加"复选框"控件

在MATLAB的控件类型中，复选框和单选按钮的功能类似，只是多个复选框控件可以同时选中。复选框为用户提供一些可以独立选择的选项进行设置程序模式，例如显示工具条与否以及生成回调函数原型与否。在本例中，复选框的功能是提供用户选择图形显示的选项，"transpose"选项的含义是将图形进行转置。



说明　对于复选框控件，属性"Max"和"Min"表示了复选框控件的两种状态。在默认情况下，Max属性数值为1，Min属性数值为0。而属性"Value"则表示了与当前状态直接联系，当控件被选中时，该属性数值为1；同时，当控件没有被选中时，属性数值为0。

step 6 添加其他的复选框控件。在本例中，提供用户选择的图形选项还有上下倒转、左右倒转，可以用和上面步骤类似的方法添加其他的复选框控件，得到的结果如图11.91所示。



图 11.91　添加其他复选框控件



说明　在上面的控件名称中，flipud表示的是将图形进行上下转置，fliplr表示的是将图形进行左右转置，对应的都是MATLAB中的相关函数。

step 7 复制上面步骤中添加的控件。根据本实例中的结果，该GUI对象共有三个坐标轴系统，显示不同截面的平面图形，因此在本步骤中需要复制上面步骤中添加的控件类型，得到的结果如图 11.92 所示。

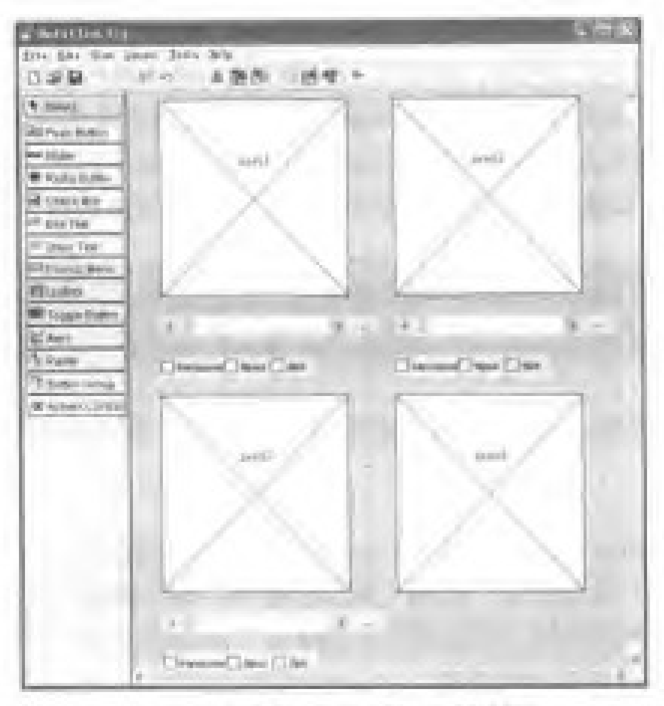


图 11.92　复制上面步骤中的控件



在上面的步骤中，除了使用简单的命令来复制前面步骤添加的控件之外，还需要使用排列工具将上面的控件进行排列，具体的使用方法请读者参见前面的章节。

step 8 添加其他的图形控件。根据本实例的要求，为实现其他图形功能，需要添加其他图形控件，具体的控件类型包括编辑框、静态文本、按钮等。这些控件类型在前面介绍过，这里就不重复介绍了，添加后的图形界面如图 11.93 所示。

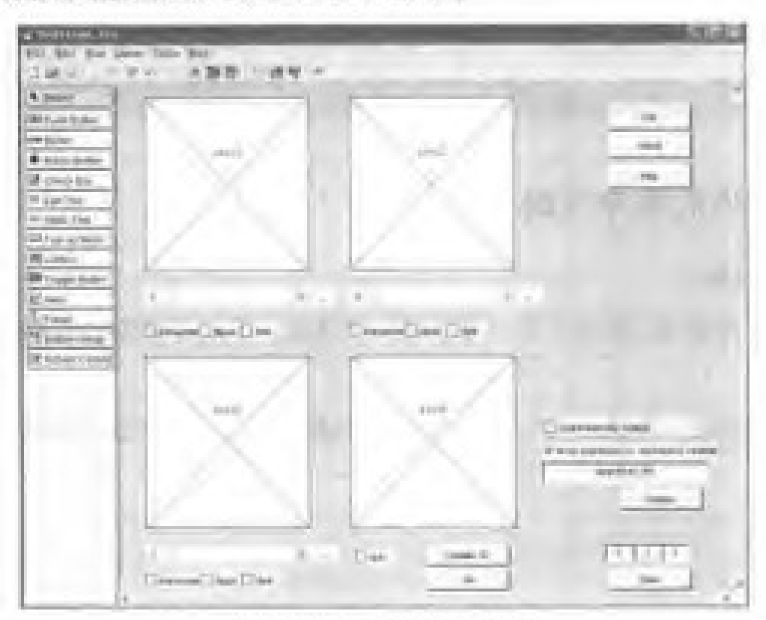


图 11.93　添加图形控件

step 9 分析图形控件的功能。

- 静态文本（Static Text）之所以被称为“静态”，是因为其中的文字只是起到注释说明的作用，而不能产生“回调”函数；但是静态文本框中的内容可以使用程序代码进行修改。
- 编辑框（Edit Text）控件，如果用户希望在其中输入多行数据行时，必须使用分号“;”或者逗号“,”来结尾，然后再回车进行换行。当某个完整的命令长度超过了物理行时，MATLAB将会任其自动回绕。用户在编辑框中输入的文字内容，会被编辑框中的“String”属性所接收，其数值是由多行命令构成的“补尾字符串矩阵”。



在MATLAB中，通常使用eval命令来解读字符串矩阵，但是该命令在解读矩阵时，是沿“列”进行解读的，应该采取适当措施进行处理。

step 10 其他常见控件的说明。由于上面实例的控件有限，对于MATLAB中的其他常见控件，下面作出简要的说明。

- **弹出式菜单（Pop-up Menu）**：弹出式菜单将打开并显示一个由其String属性定义的选项列表。当用户希望提供一些相互排斥的选项，但不希望占用空间时，弹出式菜单将十分有用。在非激活状态，弹出框中只显示选中的选项内容。该控件的“Value”属性是正整数，标志用户所选中的选项。
- **列表框（Listbox）**：和复选框类似，列表框中所列的选项不是互斥的，用户可以同时选中列表框中的多项内容。当选中某些选项后，对应的选项会变得醒目。当然，只有将列表框中的“Max”属性设置大于2时，才能允许用户选择多个选项。当用户需要选择多个选项时，需要同时按住“Ctrl”键然后使用鼠标进行点选。
- **单选按钮（Radio Button）**：单选按钮和普通按钮在执行方式上没有本质的差别，但是单选按钮通常是以组为单位，一组单选按钮之间是一种互斥的关系，每组单选按钮只能有一个按钮被选中。该选项被选中的属性由“Max”决定，不选中的时候，其属性则由“Min”决定。



对于其他的MATLAB控件，可以查看对应的MATLAB帮助信息，限于篇幅，这里不详细介绍。

11.8.2 添加控件的程序代码

延续上面小节的步骤。

step 1 查看添加的控件效果。完成上面小节的添加工作后，可以查看添加后的控件效果，如图11.94所示。

当添加所有控件后，在本小节中需要编写相应的程序代码，下面分步骤详细介绍本实例中的程序代码。



图 11.94 添加控件的效果

step 2 在 GUIDE 界面中，打开 GUI 对象的代码编辑器，然后在其中输入下面的代码：

```
% -----------------------------------------------------------------------
function myplot(handles,n)
sp1=1/str2num(get(handles.edit2,'string'));
sp2=1/str2num(get(handles.edit3,'string'));
sp3=1/str2num(get(handles.edit4,'string'));
vmx=max(handles.vol(:));
vmn=min(handles.vol(:));
s1=round(get(handles.slider1,'value'));
% 创建 x-yz 图形截面
if any(n==1)
    I=squeeze(handles.vol(s1,:,:));
    if get(handles.checkbox1,'value'), I=I'; end
    if get(handles.checkbox4,'value'), I=flipud(I); end
    if get(handles.checkbox7,'value'), I=fliplr(I); end
    %    axes(handles.axes1);imagesc(I);
    axes(handles.axes1);
    if get(handles.checkbox11,'value');imagesc(I,[ vmn,vmx]);else imagesc
(I);end
    colormap(gray)
    pbaspect('manual');pbaspect(handles.axes1,[ sp2,sp3,sp1]);
    %    c=get(handles.axes1,'Children'); set(c(1),'cdata',I);
    set(handles.text1,'string',[ 'x=',num2str(s1)])
    title('y-z')
end
s2=round(get(handles.slider2,'value'));
if any(n==2)% y -- x z
    I=squeeze(handles.vol(:,s2,:));
    if get(handles.checkbox2,'value'), I=I'; end
    if get(handles.checkbox5,'value'), I=flipud(I); end
    if get(handles.checkbox8,'value'), I=fliplr(I); end
    %    axes(handles.axes2);imagesc(I);
    axes(handles.axes2);
```

```
    if get(handles.checkbox11,'value');imagesc(I,[vmn,vmx]);else imagesc
(I);end
    colormap(gray)
    pbaspect('manual');pbaspect(handles.axes2,[sp1,sp3,sp2]);
    %    c=get(handles.axes2,'Children'); set(c(1),'cdata',I);
    set(handles.text2,'string',['y=',num2str(s2)])
    title('x-z')
end
s3=round(get(handles.slider3,'value'));
if any(n==3)% z -- x y
    I=squeeze(handles.vol(:,:,s3));
    if get(handles.checkbox3,'value'), I=I'; end
    if get(handles.checkbox6,'value'), I=flipud(I); end
    if get(handles.checkbox9,'value'), I=fliplr(I); end
    %    axes(handles.axes3); imagesc(I);
    axes(handles.axes3);
    if get(handles.checkbox11,'value');imagesc(I,[vmn,vmx]);else imagesc
(I);end
    colormap(gray)
    pbaspect('manual');pbaspect(handles.axes3,[sp1,sp2,sp3]);
    %    c=get(handles.axes3,'Children'); set(c(1),'cdata',I);
    set(handles.text3,'string',['z=',num2str(s3)])
    title('x-y')
end
if get(handles.checkbox10,'value'),
    pushbutton3_Callback([],[],handles,[]);
end
```

上面的程序代码是用户自定义的底层绘图函数，GUI中的相关控件的回调函数都将调用该函数的程序代码，其主要功能是绘制各截面的图形。

step 3 编写GUI控件的回调函数。将打开的M文件编辑器回嵌到命令窗口中，然后选择对应的控件回调函数名称，编写对应的回调函数，如图11.95所示。

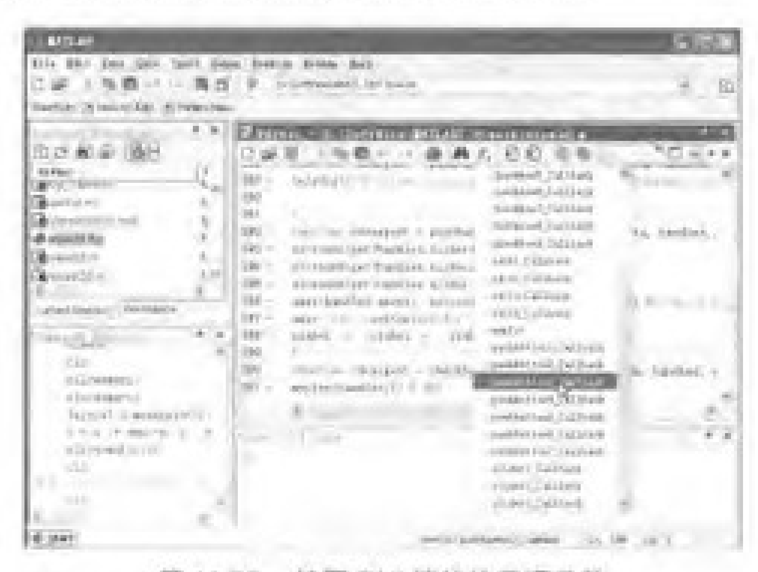


图11.95 编写GUI控件的回调函数

由于本实例中，控件的回调函数比较简单，下面显示所有回调函数的代码

```
function varargout = view3d(varargin)
```

```
%VIEW3D GUI for interactive viewing of 3D Volumes
%   VIEW3D is used view orthographic slices of 3D volumes
%   Type in an expression that generates a 3D array
%   then press the Display button
%
%   3D expressions such as: rand(50,40,30) or
%   the name or a 3D array variable in the workspace
%
%   See also: SLICE, MONTAGE, ISOSURFACE
if ~isempty(varargin) & (all(size(varargin{1})==[3 1]) | all(size(varargin
{1})==[1 3]))
    spanvar=varargin{1};
end
if nargin == 0 | exist('spanvar')% LAUNCH GUI
     fig = openfig(mfilename,'reuse');        % Generate a structure of
handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);
    if nargout > 0
        varargout{1} = fig;
    end
        if exist('spanvar')
        set(handles.edit2,'string',num2str(spanvar(1)))
        set(handles.edit3,'string',num2str(spanvar(2)))
        set(handles.edit4,'string',num2str(spanvar(3)))
    end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
    try
        if (nargout)
           [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        else
            feval(varargin{:}); % FEVAL switchyard
        end
    catch
        disp(lasterr);
    end
end
%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
function varargout = slider1_Callback(h, eventdata, handles, varargin)
myplot(handles,1);
% --------------------------------------------------------------------
function varargout = slider2_Callback(h, eventdata, handles, varargin)
myplot(handles,2);
% --------------------------------------------------------------------
function varargout = slider3_Callback(h, eventdata, handles, varargin)
myplot(handles,3);
% --------------------------------------------------------------------
function varargout = edit1_Callback(h, eventdata, handles, varargin)
% --------------------------------------------------------------------
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
a=get(handles.edit1,'String');
handles.vol=double(squeeze(evalin('base',a)));
if ndims(handles.vol)~=3,
    disp('not 3d')
```

```
    return
end
[ handles.sx,handles.sy,handles.sz] =size(handles.vol);
set(handles.slider1,'min',1);
set(handles.slider2,'min',1);
set(handles.slider3,'min',1);
set(handles.slider1,'max',handles.sx);
set(handles.slider2,'max',handles.sy);
set(handles.slider3,'max',handles.sz);
set(handles.slider1,'value',round(handles.sx/2)+1);
set(handles.slider2,'value',round(handles.sy/2)+1);
set(handles.slider3,'value',round(handles.sz/2)+1);
cla(handles.axes4);axis([ 1 handles.sx 1 handles.sy 1 handles.sz] ); axis vis3d
%axes(handles.axes1);imagesc(squeeze(handles.vol(1,:,:)));axis image;
%axes(handles.axes2);imagesc(squeeze(handles.vol(:,1,:)));axis image;
%axes(handles.axes3);imagesc(squeeze(handles.vol(:,:,1)));axis image;
set(gcf,'DoubleBuffer','on');
myplot(handles,[ 1 2 3] )
%%% produced error in matlab 7.0
%if ~isfield(handles,'clrmnu')
%    handles.clrmnu=0;
%end
%if ~handles.clrmnu;
%    colormenu;
%    handles.clrmnu=1;
%end
guidata(h,handles);
% --------------------------------------------------------------------
function varargout = checkbox1_Callback(h, eventdata, handles, varargin)
myplot(handles,1);
% --------------------------------------------------------------------
function varargout = checkbox2_Callback(h, eventdata, handles, varargin)
myplot(handles,2);
% --------------------------------------------------------------------
function varargout = checkbox3_Callback(h, eventdata, handles, varargin)
myplot(handles,3);
% --------------------------------------------------------------------
function varargout = checkbox4_Callback(h, eventdata, handles, varargin)
myplot(handles,1);
% --------------------------------------------------------------------
function varargout = checkbox5_Callback(h, eventdata, handles, varargin)
myplot(handles,2);
% --------------------------------------------------------------------
function varargout = checkbox6_Callback(h, eventdata, handles, varargin)
myplot(handles,3);
% --------------------------------------------------------------------
function varargout = checkbox7_Callback(h, eventdata, handles, varargin)
myplot(handles,1);
% --------------------------------------------------------------------
function varargout = checkbox8_Callback(h, eventdata, handles, varargin)
myplot(handles,2);
% --------------------------------------------------------------------
function varargout = checkbox9_Callback(h, eventdata, handles, varargin)
myplot(handles,3);
% --------------------------------------------------------------------
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
```

```
helpdlg({'3D Volume Orthoslice Viewer','(c) Ghassan Hamarneh 2002-2004'})
% -----------------------------------------------------------------------
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
s1=round(get(handles.slider1,'value'));
s2=round(get(handles.slider2,'value'));
s3=round(get(handles.slider3,'value'));
axes(handles.axes4); hslc=slice(handles.vol,s1,s2,s3);%rotate3d on;
axis tight; set(hslc(1:3),'LineStyle','none');
xlabel 'x' ;ylabel 'y' ;zlabel 'z';
% -----------------------------------------------------------------------
function varargout = checkbox10_Callback(h, eventdata, handles, varargin)
myplot(handles,[ 1 2 3] );
% -----------------------------------------------------------------------
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
cla(handles.axes4);
% -----------------------------------------------------------------------
function varargout = edit2_Callback(h, eventdata, handles, varargin)
myplot(handles,[ 1 2 3] );
% -----------------------------------------------------------------------
function varargout = edit3_Callback(h, eventdata, handles, varargin)
myplot(handles,[ 1 2 3] );
% -----------------------------------------------------------------------
function varargout = edit4_Callback(h, eventdata, handles, varargin)
myplot(handles,[ 1 2 3] );
% -----------------------------------------------------------------------
function pushbutton5_Callback(hObject, eventdata, handles)
helpdlg({'- Type in an expression that generates a 3D array              ',...
        '   then press the Load  button                                   ',...
        '- 3D expressions such as: rand(50,40,30) or                      ',...
        '    the name or a 3D array variable in the workspace             ',...
        '- The 3 views (all except the lower right one)                   ',...
        '    display orthographic projections                             ',...
        '- Use the scroll bars to change the number of the slice viewed    '...
        '- Use the transpose, flipud, or fliplr to                         '...
        '    transpose the view, flip it vertically, or horizontally      ',...
        '- Use update 3d to view the slices in a 3D view                  ',...
        '- Check auto to obtain an automatic update of the 3D view of the slices',...
        '    (Note: this may affect performance)                          ',...
        '- Use cla to clear the 3D view                                   ',...
        '    this may improve parformance                                 ',...
        '- Change the span values to the volume''s physical dimensions    ',...
        '    so the aspect ratio is displayed properly                    ',...
        '     (note: you can use relative values                          ',...
        '    for example use 1,3,2 instead of 0.5,1.5,1.0)                ',...
        '     then press Apply                                            '})
function pushbutton6_Callback(hObject, eventdata, handles)
if strcmp(questdlg('Exit View3D?','View3D','Yes','No','No'),'Yes')
    close(handles.view3d);
end
function pushbutton7_Callback(hObject, eventdata, handles)
myplot(handles,[ 1 2 3] );
```

运行程序代码

延续上面小节的步骤。

step 1 查看默认的程序代码结果。在 MATLAB 的命令窗口中输入 ">>view3d"，按 "Enter" 键，得出默认的程序结果，如图 11.96 所示。

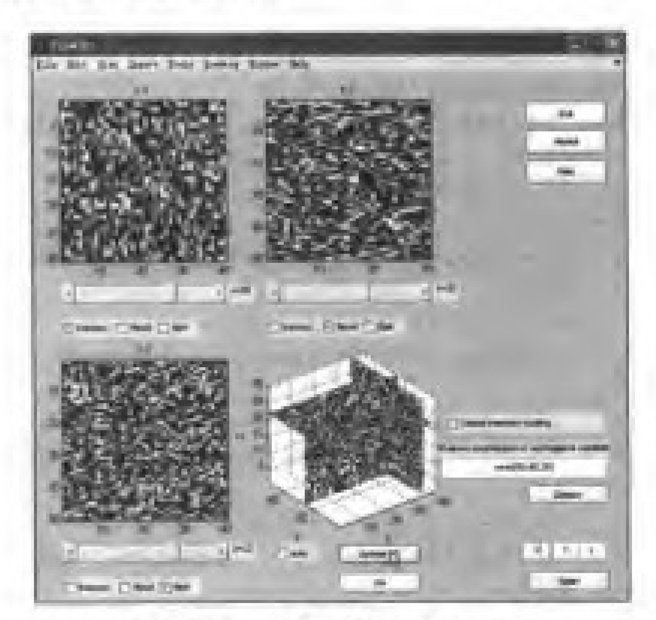


图 11.96　查看默认的程序结果

在默认情况下，绘制的三维数组是 "rand(50,40,30)"，同时选择各个截面的坐标数值和图形选项，得到的三维图形在右下方的坐标轴系统中显示。

step 2 返回 M 文件编辑器中，输入下面的程序代码：

```
>> x=linspace(-3*pi,3*pi,1000);
>>y=x;
>> [X,Y]=meshgrid(x,y);
>>R=sqrt(X.^2+Y.^2)+eps;
>>Z=sin(R)./R;
>>A3D=reshape(Z,100,100,[]);
>>view3d
```

step 3 查看图形结果。输入代码后，按 "Enter" 键，得到的结果如图 11.97 所示。

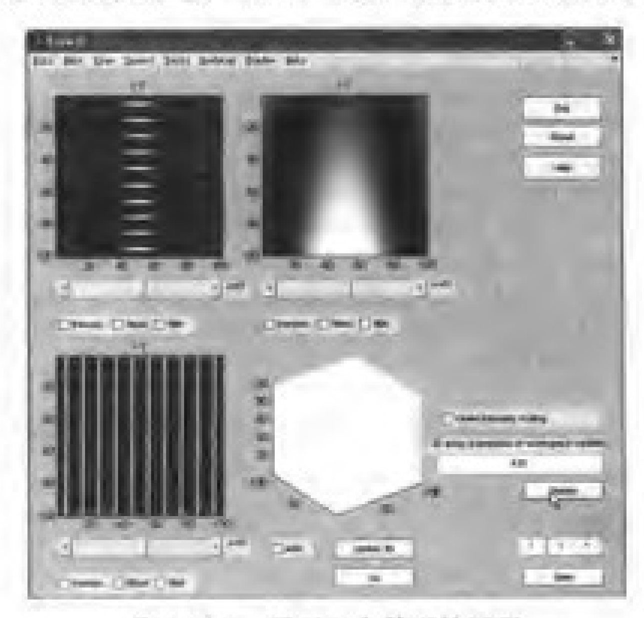


图 11.97　显示三个截面的图形

在上面的程序中，用户定义了一个三维数组 A3D，然后在图形界面中单击 "Display" 按钮，就会显示对应的截面图。

step 4 显示三维图形。单击上面图形界面中的 "Update 3D" 按钮，得到的图形结果如图 11.98 所示。

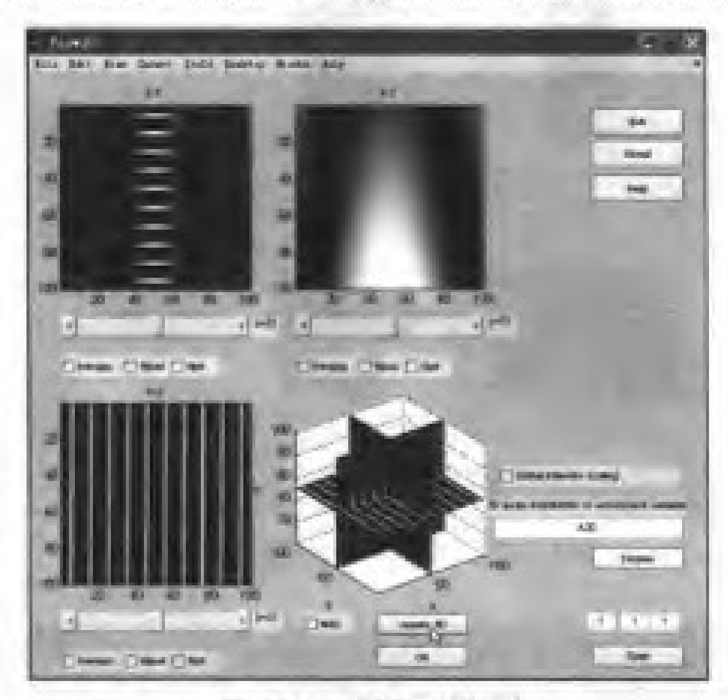


图 11.98　显示三维图形

step 5 修改截面属性，重新查看图形结果。对于 y-z 截面，将其坐标数值设置为 "x = 70"，同时选中 "flipud" 选项，对于 x-z 截面，将其坐标数值设置为 "y = 30"，同时选中 "transpose" 选项；对于 x-y 截面，将其坐标数值设置为 "z = 60"，同时选中 "fliplr" 选项。设置这些属性选项后，单击 "Update 3D" 选项，得到的结果如图 11.99 所示。

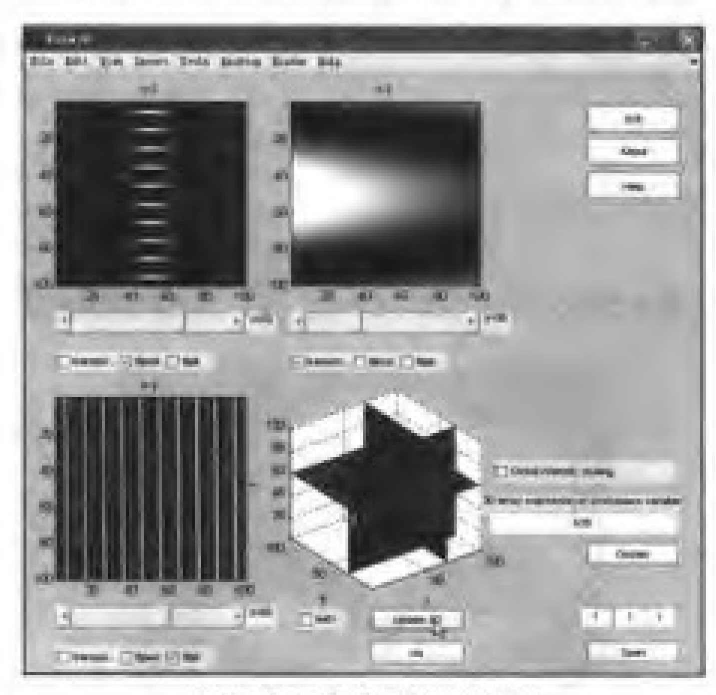


图 11.99　修改后的图形界面

step 6 修改显示比例。在文本框中输入图形窗口显示比例，得到的图形如图 11.100 所示。

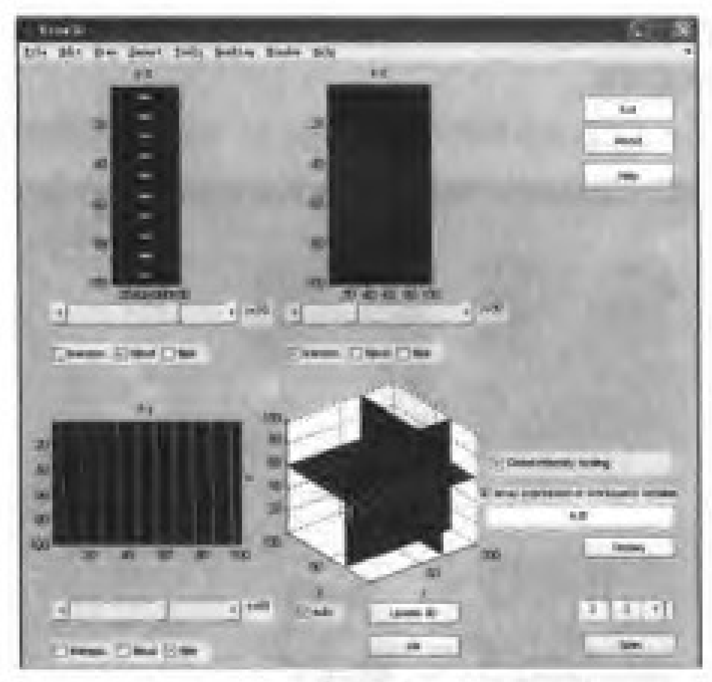


图 11.100　修改图形窗口的显示比例



在上面的程序步骤中，已经基本演示了该GUI对象的主要功能，表明该GUI程序代码设置的成功。读者可以使用其他三维数组来检测GUI对象的功能，这里就不重复介绍了。

step 7 查看三维图形。为了让读者更加直观地了解上面图形的结构，可以在MATLAB的命令窗口中输入如下代码：

```
>> x=linspace(-3*pi,3*pi,100);
>>y=x;
>> [X,Y]=meshgrid(x,y);
>>R=sqrt(X.^2+Y.^2)+eps;
>>Z=sin(R)./R;
>>surf(X,Y,Z)
>>shading interp ;colormap hsv
>>colorbar
```

step 8 查看图形结果。输入程序代码后，按"Enter"键，得到的结果如图 11.101 所示。

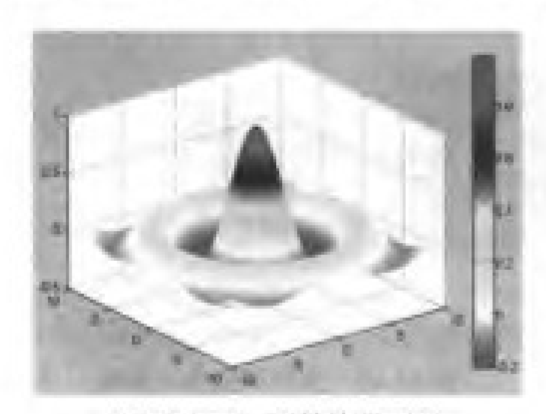


图 11.101　函数的曲面图

在上面的GUI对话框中，显示的是上面曲面图形的三个垂直截面图形结果，尽管截面的图形结果也能显示数值变化趋势，但是曲面图更加直观地显示了数据变化的结果。

11.9 综合案例

前面介绍了如何在MATLAB中创建GUI对象的方法，并介绍了如何创建GUI菜单、工具栏、控件的具体方法和注意事项。本节将介绍一个比较综合的案例，说明如何综合利用这些方法创建一个比较复杂的GUI对象。

具体来讲，该GUI对象的主要功能是动态演示绘制三元函数的切片图的过程，以某个三维数组为例，得到的切片图如图11.102所示。

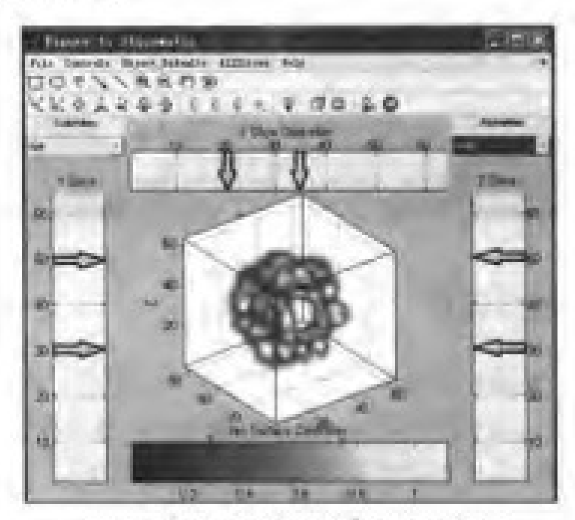


图 11.102 完成的 GUI 图形界面

11.9.1 分析 GUI 对象

由于该GUI对象比较复杂，因此有必要认真地分析GUI对象的各种功能和性能。首先，从GUI组件的角度分析，该GUI对象包括自定义GUI菜单选项、自定义GUI工具栏、自定义GUI用户控件等组件；从用户和计算机的互动角度分析，该GUI包括鼠标互动、控件按钮互动等。



该GUI对象的鼠标互动情况和前面章节中的鼠标互动不同，在本GUI对象中，除了互动之外，还向该GUI对象中添加了其他图形对象——箭头。

下面详细分析该GUI中的各种对象的具体属性。

✦ 自定义菜单选项

前面介绍过在MATLAB中创建菜单选项的方法，在本GUI对象中，菜单选项还包括"help"菜单选项。该菜单选项会启动MATLAB的帮助系统，对于该选项的创建方法将在后面章节中详细介绍。

✦ 自定义工具栏

在本GUI对象中，工具栏包括常见的绘图工具选项和照明工具栏，而不是MATLAB中的默认菜单工具栏。之所以选择该工具栏选项，是因为这些选项是查看该GUI中图形对象的常用工具选项。

✦ 自定义用户控件

在本GUI对象中，用户控件包括两个下拉菜单，分别用来选择图形的色图（colormap）系统和透

明度（Alphamap）选项，这些选项都将直接影响图形对象的显示结果。

下面再详细介绍该GUI对象的操作步骤。

- 首先在MATLAB的工作空间中创建一个三维数组[X,Y,Z]和一个同维向量V，然后输入命令sliceomatic(V)或者sliceomatic(V,X,Y,Z)，调用该GUI对象。
- 通过鼠标来确定该三维图形的切面的坐标数值，包括X向、Y向和Z向的切面坐标数值，该GUI允许用户在各个方向上分别确定多个切面。当选择某个切面后，MATLAB会显示对应坐标条件下的切面图形。
- 可以分别在两个下拉菜单中选择色图系统和图形透明度数值，来改变图形显示的颜色和透明度。
- 可以使用照明工具栏中对应的按钮，修改或查看三维切面图形的视角。
- 可以使用菜单选项来设置图形显示的各种属性。
- 可以使用help来查看关于该命令的帮助信息。



从上面的分析中可以看出，该GUI对象几乎集合了所有关于图形属性的内容，比较复杂，因此需要分章节详细介绍具体的创建过程。

11.9.2 规划GUI的设计过程

根据前面的介绍，本实例整体比较复杂，涉及的对象比较繁多。为了有效地创建整个GUI对象，在创建该GUI之前，必须对创建过程进行规划。为了设置GUI中各个控件的属性，需要为创建控件编写对应的M文件，同时，为调用各控件函数编写主函数的程序代码。

为了显示整个系统的完整性，将所有被调函数的M文件保存在"private"文件夹中，主调函数则是保存在主文件中，下面分别介绍如何创建各种控件。

11.9.3 创建GUI的工具栏对象

从本小节开始，将介绍如何在MATLAB中创建该GUI对象。

例11.8 创建11.9.1小节中的GUI对象。

step 1 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function outd = figtoolbar(d)
% 创建图形的工具栏对象
% 清除当前图形窗口的工具栏对象
  set(gcf,'toolbar','none');
if exist('uitoolfactory') == 2
    % 创建对于该GUI对象有效的工具栏选项
    d.toolbar = uitoolbar('parent',gcf);
    uitoolfactory(d.toolbar, 'Annotation.InsertRectangle');
    uitoolfactory(d.toolbar, 'Annotation.InsertEllipse');
    uitoolfactory(d.toolbar, 'Annotation.InsertTextbox');
    uitoolfactory(d.toolbar, 'Annotation.InsertArrow');
    uitoolfactory(d.toolbar, 'Annotation.InsertLine');
    uitoolfactory(d.toolbar, 'Exploration.ZoomIn');
    uitoolfactory(d.toolbar, 'Exploration.ZoomOut');
```

```
    uitoolfactory(d.toolbar, 'Exploration.Pan');
uitoolfactory(d.toolbar, 'Exploration.Rotate');
% 设置工具栏的照明属性
    cameratoolbar('show');
    cameratoolbar('togglescenelight');
      else
    % 对R13或者更早的版本使用下面的程序代码
    try
      cameratoolbar('show');
      cameratoolbar('togglescenelight');
      %cameratoolbar('setmode','orbit');
    catch
      disp('Could not display the camera toolbar.');
    end
      end
    outd = d;
```

在上面的程序代码中，首先使用set命令将MATLAB中默认的图形窗口工具栏设置为"none"，然后使用uitoolfactory命令添加MATLAB中的默认工具栏按钮，主要有注释（annotation）和视角（exploration）两种。最后，使用cameratoolbar添加相关的照明工具按钮。上面程序的主要功能是为设置sliceomatic对象属性选择对应的工具栏按钮。



在上面程序代码的最后部分，考虑了MATLAB版本兼容的问题。如果用户使用的MATLAB版本是R13或者更新的版本，则调用最后的程序代码。

step 2 查看完成的图形界面。将上面的程序代码保存为"figtoolbar.m"文件，然后在命令窗口中输入"figtoolbar"，按"Enter"键，得到的图形如图11.103所示。

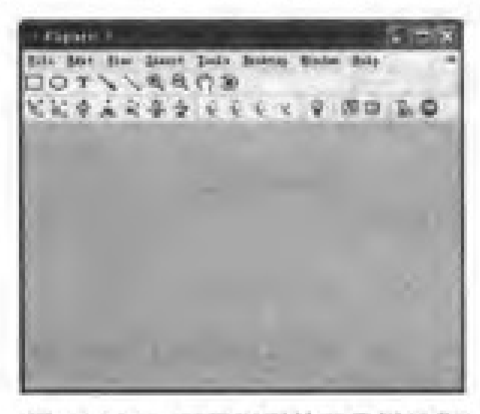


图 11.103　显示图形的工具栏对象

11.9.4 准备图形对象的基础文件

延续上面小节的步骤。

step 1 单击命令窗口工具栏中的按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function appdata = sliceomaticsetdata(d,xmesh,ymesh,zmesh)
% SLICEOMATICSETDATA(rawdata) - Create the data used for
```

```
% sliceomatic in the appdata D.
% Check variables
  error(nargchk(1,4,nargin))
  % Simplify the isonormals
  disp('Smoothing for IsoNormals...');
  d.smooth=smooth3(d.data);
  d.reducenumbers=[ floor(size(d.data,2)/20)...
                    floor(size(d.data,1)/20)...
                    floor(size(d.data,3)/20) ];
  d.reducenumbers(d.reducenumbers==0)=1;
  if nargin == 4
    % Reorder vectors: make them horizontal (prepare to flipdim)
    if size(xmesh,1)>size(xmesh,2)
      xmesh=xmesh';
    end
    if size(ymesh,1)>size(ymesh,2)
      ymesh=ymesh';
    end
    if size(zmesh,1)>size(zmesh,2)
      zmesh=zmesh';
    end
    % Set axis orientation
    xdir='normal';
    ydir='normal';
    zdir='normal';
    if issorted(xmesh)~=1
      xmesh=flipdim(xmesh,2);
      xdir='reverse';
      d.xlim = [ xmesh(1) xmesh(end)];
      xmesh=flipdim(xmesh,2);
    else
      d.xlim = [ xmesh(1) xmesh(end)];
    end
    if issorted(ymesh)~=1
      ymesh=flipdim(ymesh,2);
      ydir='reverse';
      d.ylim = [ ymesh(1) ymesh(end)];
      ymesh=flipdim(ymesh,2);
    else
      d.ylim = [ ymesh(1) ymesh(end)];
    end
    % This should not be the case for medical images
    if issorted(zmesh)~=1
      zmesh=flipdim(zmesh,2);
      zdir='reverse';
      d.zlim = [ zmesh(1) zmesh(end)];
      zmesh=flipdim(zmesh,2);
    else
      d.zlim = [ zmesh(1) zmesh(end)];
    end

    % Vol vis suite takes numbers in X/Y form.
    ly = 1:d.reducenumbers(1):size(d.data,2);
    lx = 1:d.reducenumbers(2):size(d.data,1);
    lz = 1:d.reducenumbers(3):size(d.data,3);
```

```
  for i = 1:length(ly)
    ly(i) = xmesh(ly(i));
  end
  for i = 1:length(lx)
    lx(i) = ymesh(lx(i));
  end
  for i = 1:length(lz)
    lz(i) = zmesh(lz(i));
  end

  d.reducelims={ ly lx lz };
  disp('Generating reduction volume...');
  d.reduce= reducevolume(d.data,d.reducenumbers);
  d.reducesmooth=smooth3(d.reduce,'box',5);
  % Set axis
  %d.xlim = [ xmesh(1) xmesh(end)];
  %d.ylim = [ ymesh(1) ymesh(end)];
  %d.zlim = [ zmesh(1) zmesh(end)];
  d.xmesh = xmesh;
  d.ymesh = ymesh;
  d.zmesh = zmesh;
  d.xdir = xdir;
  d.ydir = ydir;
  d.zdir = zdir;

else
  % Vol vis suite takes numbers in X/Y form.
  ly = 1:d.reducenumbers(1):size(d.data,2);
  lx = 1:d.reducenumbers(2):size(d.data,1);
  lz = 1:d.reducenumbers(3):size(d.data,3);

  d.reducelims={ ly lx lz };
  disp('Generating reduction volume...');
  d.reduce= reducevolume(d.data,d.reducenumbers);
  d.reducesmooth=smooth3(d.reduce,'box',5);

  d.xlim = [1 size(d.data,2)];
  d.ylim = [1 size(d.data,1)];
  d.zlim = [1 size(d.data,3)];
  d.xmesh = nan;
  d.ymesh = nan;
  d.zmesh = nan;
  d.xdir = 'normal';
  d.ydir = 'normal';
  d.zdir = 'normal';
end

appdata = d;
```

上面的程序代码比较复杂，下面详细介绍其具体的含义。

- 程序代码error(nargchk(1,4,nargin))……d.reducenumbers(d.reducenumbers==0)=1 的功能是判断函数的参数个数，然后将 isonormal 的数据进行归一处理。
- 程序代码 if nargin == 4……zmesh=zmesh', end 的功能是将函数的参数进行转置，将列向量转换成为行向量，这个步骤的主要目的在于为后面图形转换作准备。

- 程序代码 xdir='normal'……d.zlim = [zmesh(1) zmesh(end)]，end 的功能是，首先设置三向坐标系统的方向为正常方向，然后使用三个循环结构将数据按照图形对象结构进行转换，并转换坐标轴方向，设置坐标轴的刻度范围。
- 程序代码 ly = 1:d.reducenumbers(1):size(d.data,2) ……d.zdir = zdir 的功能是，将三维数据转换成为 X/Y 坐标轴中的数据系列，然后重新设置坐标轴的方向。
- 程序代码 else……appdata = d 的功能和上面的程序代码段类似，只是处理的情况是函数的参数个数不是 4 个的情况，原理相同，这里就不重复介绍了。

将上面的程序代码保存为"sliceomaticsetdata.m"文件，该文件将是在后面的步骤中绘制三维图形的基础文件。

step 2 单击命令窗口工具栏中的 按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file" 命令，打开一个空白的 M 文件编辑器，然后在 M 文件编辑器中输入下面的代码：

```
function sliceomaticmotion(fig,action)
% Handle generic motion events for the figure window.
  obj = hittest(fig);

  % 当用户使用鼠标在图形窗口中移动时显示光标
  if ~isempty(obj)
    t = getappdata(obj,'motionpointer');
    cc = get(fig,'pointer');
      if t
      newc = t;
    else
      newc = get(0,'defaultfigurepointer');
    end
      if isa(newc,'char') && isa(cc,'char') && ~strcmp(newc,cc)
      setpointer(fig, newc);
    end
  end
    d = getappdata(fig,'sliceomatic');
  % 创建切面直线
  if isempty(d.motionmetaslice)
    d.motionmetaslice = line('parent',d.axmain,...
                             'vis','off',...
                             'linestyle','--',...
                             'marker','none',...
                             'linewidth',2,...
                             'erasemode','xor','clipping','off');
    setappdata(fig,'sliceomatic',d);
  end
  showarrowtip(obj);
    if isempty(obj) || (obj ~= d.axx && obj ~= d.axy && obj ~= d.axz)
    set(d.motionmetaslice,'visible','off');
       return
  end

  aa = obj;
  apos=get(aa,'currentpoint');
  xl = d.xlim;
  yl = d.ylim;
  zl = d.zlim;
```

```
% 获取鼠标移动处的数值坐标
    if aa==d.axx || aa==d.axiso
    if aa==d.axiso
         else
      xdata = [ apos(1,1) apos(1,1) apos(1,1) apos(1,1) apos(1,1) ];
      ydata = [ yl(1) yl(2) yl(2) yl(1) yl(1) ];
      zdata = [ zl(2) zl(2) zl(1) zl(1) zl(2) ];
    end
  else
    % 用户绘制 Y 向或者 Z 向的切片图的坐标
    if aa==d.axy
      ydata = [ apos(1,2) apos(1,2) apos(1,2) apos(1,2) apos(1,2) ];
      xdata = [ xl(1) xl(2) xl(2) xl(1) xl(1) ];
      zdata = [ zl(2) zl(2) zl(1) zl(1) zl(2) ];
    else
      zdata = [ apos(1,2) apos(1,2) apos(1,2) apos(1,2) apos(1,2) ];
      ydata = [ yl(1) yl(2) yl(2) yl(1) yl(1) ];
      xdata = [ xl(2) xl(2) xl(1) xl(1) xl(2) ];
    end
  end
  set(d.motionmetaslice,'visible','on',...
                    'xdata',xdata,'ydata',ydata,'zdata',zdata);
```

上面程序代码的主要功能是处理鼠标在图形对象中移动时的问题。根据最后的GUI结果要求，当鼠标在各个坐标轴Slice控制面板中移动时，应该显示出对应的光标符号，同时在图形窗口中显示对应的截面。将上面的程序代码保存为"sliceomaticmotion.m"文件，该文件也将是在后面的步骤中绘制三维图形的基础文件。



由于鼠标移动的事件必定涉及鼠标光标、箭头符号等对象，因此，在上面的程序代码中，调用了setpointer、showarrowtip等相关函数，这些函数都将在后面详细介绍。

step 3 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function activelabel(label, string)
% ACTIVELABEL(LABEL, STRING) - Create a label on GCA which is
%     active.  LABEL is the property of GCA whose label you are
%     setting.  STRING is the initial text string for the label.
  l = get(gca,label);
    set(l,'string',string);
  set(l,'buttondownfcn',@activelabelbuttondown);
function activelabelbuttondown(obj, action)
% Callback when one of our active labels is clicked on.
    set(obj,'edit','on');
```

上面的程序代码功能在于设置图形各对象标签的属性，主要在于设置标签的名称和启动标签编辑的功能。将上面的程序代码保存为"activelabel.m"，该M文件将在后面的程序代码中反复被调用。



上面的程序代码比较简单，但是和前面的程序代码有很大的区别。前面的程序代码都是实现独立功能的，而上面程序代码的功能是创建被调用函数。

step 4 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function slicecontrols(fig,onoff,xmesh,ymesh,zmesh,xdir,ydir,zdir)
% 检查变量属性
  error(nargchk(2,8,nargin))
% 返回图形窗口的应用程序数据
  d = getappdata(fig, 'sliceomatic');
    if onoff
       if nargin ~= 8
           % 如果用户没有指定mesh对象，创建该对象
      xmesh(1) = 1;
      xmesh(2) = size(d.data,2);
      ymesh(1) = 1;
      ymesh(2) = size(d.data,1);
      zmesh(1) = 1;
      zmesh(2) = size(d.data,3);
% 设置图形坐标轴的方向
      xdir = 'normal';
      ydir = 'normal';
      zdir = 'normal';
end
% 创建图形窗口，并添加对应的图形数据
    set(0,'currentfigure',fig);
set([ d.axx d.axy d.axz] ,'handlevisibility','on');
% 设置图形窗口的坐标轴属性
        set(fig,'currentaxes',d.axx);
    set(d.axx, 'xlim',[ xmesh(1) xmesh(end)] ,...
               'ylim',[ 1 5] );
% 设置图形窗口中"Slice"控制器对象的属性
    set(d.pxx, 'vertices',[  xmesh(1) xmesh(1) -1; xmesh(end) xmesh(1)
-1; xmesh(end) 5 -1; xmesh(1) 5 -1] ,...
               'faces',[  1 2 3 ; 1 3 4] );
        activelabel('title', 'X Slice Controller');
        set(fig,'currentaxes',d.axy);
% 设置xy截面切片图的控件属性
    set(d.axy, 'xlim',[ 1 5] ,...
               'ylim',[ ymesh(1) ymesh(end)] );
    set(d.pxy, 'vertices',[  ymesh(1) ymesh(1) -1; ymesh(1) ymesh(end)
-1; 5 ymesh(end) -1; 5 ymesh(1) -1] ,...
               'faces',[  1 2 3 ; 1 3 4] );
activelabel('title', 'Y Slice');
% 设置xz截面切片图的控件属性
    set(fig,'currentaxes',d.axz);
    set(d.axz, 'xlim',[ 1 5] ,...
               'ylim',[ zmesh(1) zmesh(end)] );
    set(d.pxz, 'vertices',[  zmesh(1) zmesh(1) -1; zmesh(1) zmesh(end)
-1; 5 zmesh(end) -1; 5 zmesh(1) -1] ,...
               'faces',[  1 2 3 ; 1 3 4] );
    activelabel('title', 'Z Slice');
    set([ d.axx d.axy d.axz] ,'handlevisibility','off');
```

```
    set(d.axx,'xdir',xdir);
    set(d.axy,'ydir',ydir);
    set(d.axz,'zdir',zdir);
  else

      end
```

上面的程序代码功能是创建图形界面的Slice控件。该GUI对象中包含三个Slice控件对象，用来选择三维图形对象的三个切片面的坐标数值。上面的程序代码比较简单，只是涉及一些图形句柄的语句，这里就不重复介绍了。

将上面的程序代码保存为“slicecontrols.m”，该程序代码的主要功能就是创建图形界面中的Slice控件对象。

step 5 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function isocontrols(fig, onoff)
% 检测输入变量的个数
  error(nargchk(2,2,nargin))
  d = getappdata(fig, 'sliceomatic');
    if onoff
    lim=[ min(min(min(d.data))) max(max(max(d.data)))];
     % 设置 iso控件的属性
    set(d.axiso,'handlevisibility','on');
    set(fig,'currentaxes',d.axiso);
    set(d.axiso, 'xlim',lim,...
                 'ylim',[ 1 5],...
                 'clim',lim);
% 创建图形对象
    image('parent',d.axiso,'cdata',1:64,'cdatamapping','direct',...
          'xdata',lim,'ydata',[ 0 5],...
          'alphadata',.6, ...
          'hittest','off');
    activelabel('title','Iso Surface Controller');
    set(d.axiso,'handlevisibility','off');
      else
    % 禁止 iso 控件的功能
     delete(findobj(d.axis,'type','image'));
  end
```

上面的程序代码功能是在图形界面的底部创建一个关于ISO的控件。当用户选择该控件中的某个数值的时候，可以设置图形界面中的ISO的属性值。将上面的程序代码保存为“isocontrols.m”文件，程序代码将在后面步骤中被调用。

step 6 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function appdata=sliceomaticfigure(d,xmesh,ymesh,zmesh)
% 检测输入变量的个数
error(nargchk(1,4,nargin))
% Init sliceomatic
  figure('name','Slicematic','toolbar','none');
  lim=[ min(min(min(d.data))) max(max(max(d.data)))];
```

```
if nargin==4
    % 向量转换
    if size(xmesh,1)>size(xmesh,2)
        xmesh=xmesh';
    end
    if size(ymesh,1)>size(ymesh,2)
        ymesh=ymesh';
    end
    if size(zmesh,1)>size(zmesh,2)
        zmesh=zmesh';
    end
    % 设置坐标轴的方向
    xdir='normal';
    ydir='normal';
    zdir='normal';
    if issorted(xmesh)~=1
        xmesh=flipdim(xmesh,2);
        xdir='reverse';
    end
    if issorted(ymesh)~=1
        ymesh=flipdim(ymesh,2);
        ydir='reverse';
    end
        if issorted(zmesh)~=1
        zmesh=flipdim(zmesh,2);
        zdir='reverse';
    end
    % 更新图形的数据
    d.axmain = axes('units','normal','pos',[ .2  .2 .6 .6],'box','on',...
        'ylim',[ ymesh(1) ymesh(end)] ,...
        'xlim',[ xmesh(1) xmesh(end)] ,...
        'zlim',[ zmesh(1) zmesh(end)] ,...
        'clim',lim,...
        'alim',lim);
    % 设置坐标轴的方向
    set(gca,'XDir',xdir,'YDir',ydir,'ZDir',zdir);
  else
    d.axmain = axes('units','normal','pos',[ .2  .2 .6 .6],'box','on',...
        'ylim',[ 1 size(d.data,1)] ,...
        'xlim',[ 1 size(d.data,2)] ,...
        'zlim',[ 1 size(d.data,3)] ,...
        'clim',lim,...
        'alim',lim);
end
%设置图形的坐标轴标签
activelabel('xlabel', 'X');
activelabel('ylabel', 'Y');
activelabel('zlabel', 'Z');
%activelabel('title', 'Data');
%设置图形的视角
daspect([ 1 1 1] );
view(3);
axis tight vis3d;
hold on;
grid on;
  % 依次创建四个图形控件
```

```
d.axx    = axes('units','normal','pos',[ .2  .81 .6 .1],'box','on',...
              'ytick',[],'xgrid','on','xaxislocation','top',...
              'zlim',[ -2 1 ],...
              'layer','top',...
              'color','none');
d.pxx    = patch('facecolor',[1 1 1],...
               'facealpha',.6,...
               'edgecolor','none',...
               'hittest','off');
setappdata(d.axx,'motionpointer','SOM bottom');
d.axy    = axes('units','normal','pos',[ .05 .05 .1 .75],'box','on',...
              'xtick',[],'ygrid','on',...
              'zlim',[ -2 1 ],...
              'layer','top',...
              'color','none');
d.pxy    = patch('facecolor',[1 1 1],...
               'facealpha',.6,...
               'edgecolor','none',...
               'hittest','off');
setappdata(d.axy,'motionpointer','SOM right');
d.axz    = axes('units','normal','pos',[ .85 .05 .1 .75],'box','on',...
              'xtick',[],'ygrid','on','yaxislocation','right',...
              'zlim',[ -2 1 ],...
              'layer','top',...
              'color','none');
d.pxz    = patch('facecolor',[1 1 1],...
               'facealpha',.6,...
               'edgecolor','none',...
               'hittest','off');
setappdata(d.axz,'motionpointer','SOM left');
d.axiso  = axes('units','normal','pos',[ .2 .05 .6 .1],'box','on',...
              'ytick',[],'xgrid','off','ygrid','off',...
              'xaxislocation','bottom',...
              'zlim',[ -1 1],...
              'color','none',...
              'layer','top');
setappdata(d.axiso,'motionpointer','SOM top');
set([ d.axx d.axy d.axz d.axiso],'handlevisibility','off');
setappdata(gcf,'sliceomatic',d);
% 创建默认的 sliceomatic控件
if nargin == 4
    slicecontrols(gcf,1,xmesh,ymesh,zmesh,xdir,ydir,zdir);
else
    slicecontrols(gcf,1);
end
    isocontrols(gcf,1);
%设置各个控件的回调函数
set(d.axx,'buttondownfcn','sliceomatic Xnew');
set(d.axy,'buttondownfcn','sliceomatic Ynew');
set(d.axz,'buttondownfcn','sliceomatic Znew');
set(d.axiso,'buttondownfcn','sliceomatic ISO');
% 设置鼠标移动的回调函数
d.motionmetaslice = [];
set(gcf,'windowbuttonmotionfcn',@sliceomaticmotion);
  % 创建工具栏
d=figtoolbar(d);
```

```
    d = figmenus(d);

    % 对图形窗口进行颜色和透明设置
    uicontrol('style','text','string','ColorMap',...
              'units','normal','pos',[0 .9 .19 .1]);
    uicontrol('style','popup','string',...
              {'jet','hsv','cool','hot','pink','bone','copper','flag',
  'prism','rand','custom'},...
              'callback','sliceomatic colormap',...
              'units','normal','pos',[0 .85 .19 .1]);
    uicontrol('style','text','string','AlphaMap',...
              'units','normal','pos',[.81 .9 .19 .1]);
    uicontrol('style','popup','string',{'rampup','rampdown','vup','vdown','rand'},...
              'callback','sliceomatic alphamap',...
              'units','normal','pos',[.81 .85 .19 .1]);
  %设置文本属性
    d.tip = text('visible','off','fontname','helvetica','fontsize',10,
  'color','black');
    try
        set(d.tip,'backgroundcolor',[1 1 .8],'edgecolor',[.5 .5 .5],
  'margin',5);
    end
  appdata = d;
```

上面的程序代码的主要功能是，依次创建各种图形对象，包括Slice控件、ISO控件、下拉菜单选项等，该程序代码将是绘制Slice对象的最主要的程序内容。最后，将上面的程序代码保存为"setvolumerange.m"文件。

处理指针对象

延续上面小节的步骤。

step 1 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function setpointer(fig, ptr)
% Set the pointer on the current figure to PTR
% has several specialized SOM (SliceOMatic) pointers
  switch ptr
   case 'SOM left'
    pd = [ nan nan nan nan 1   nan nan nan nan nan nan nan nan nan nan nan
           nan nan nan 1   1   nan nan nan nan nan nan nan nan nan nan nan
           nan nan nan 1   1   nan nan nan nan nan nan nan nan nan nan nan
           nan nan 1   2   1   nan nan nan nan nan nan nan nan nan nan nan
           nan nan 1   2   1   1   1   1   1   1   1   1   1   1   1   1
           nan 1   2   2   2   2   2   2   2   2   2   2   2   2   2   1
           nan 1   2   2   2   2   2   2   2   2   2   2   2   2   2   1
           1   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1
           1   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1
           nan 1   2   2   2   2   2   2   2   2   2   2   2   2   2   1
           nan 1   2   2   2   2   2   2   2   2   2   2   2   2   2   1
           nan nan 1   2   1   1   1   1   1   1   1   1   1   1   1   1
```

```
        nan nan 1   2   1   nan nan nan nan nan nan nan nan nan nan nan
        nan nan nan 1   1   nan nan nan nan nan nan nan nan nan nan nan
        nan nan nan 1   1   nan nan nan nan nan nan nan nan nan nan nan
        nan nan nan nan 1   nan nan nan nan nan nan nan nan nan nan nan ];
 set(fig,'pointershapecdata', pd,...
       'pointershapehotspot', [ 8 1 ] , ...
       'pointer','custom');
case 'SOM right'
 pd = [ nan nan nan nan nan nan nan nan nan nan nan 1   nan nan nan nan
        nan nan nan nan nan nan nan nan nan nan nan 1   1   nan nan nan
        nan nan nan nan nan nan nan nan nan nan nan 1   1   nan nan nan
        nan nan nan nan nan nan nan nan nan nan nan 1   2   1   nan nan
        1   1   1   1   1   1   1   1   1   1   1   1   2   1   nan nan
        1   2   2   2   2   2   2   2   2   2   2   2   2   2   1   nan
        1   2   2   2   2   2   2   2   2   2   2   2   2   2   1   nan
        1   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1
        1   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1
        1   2   2   2   2   2   2   2   2   2   2   2   2   2   1   nan
        1   2   2   2   2   2   2   2   2   2   2   2   2   2   1   nan
        1   1   1   1   1   1   1   1   1   1   1   1   2   1   nan nan
        nan nan nan nan nan nan nan nan nan nan nan 1   2   1   nan nan
        nan nan nan nan nan nan nan nan nan nan nan 1   1   nan nan nan
        nan nan nan nan nan nan nan nan nan nan nan 1   1   nan nan nan
        nan nan nan nan nan nan nan nan nan nan nan 1   nan nan nan nan ];
 set(fig,'pointershapecdata', pd,...
       'pointershapehotspot', [ 8 16 ] , ...
       'pointer','custom');
case 'SOM bottom'
 pd = [ nan nan nan nan 1   1   1   1 1 1   1   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        1   1   1   1   1   2   2   2 2 2   2   1   1   1   1   1
        nan 1   1   2   2   2   2   2 2 2   2   2   2   1   1   nan
        nan nan nan 1   1   2   2   2 2 2   2   1   1   nan nan nan
        nan nan nan nan nan 1   1   2 2 1   1   nan nan nan nan nan
        nan nan nan nan nan nan nan 1 1 nan nan nan nan nan nan nan ];
 set(fig,'pointershapecdata', pd,...
       'pointershapehotspot', [ 16 8 ] , ...
       'pointer','custom');
  case 'SOM top'
 pd = [ nan nan nan nan nan nan nan 1 1 nan nan nan nan nan nan nan
        nan nan nan nan nan 1   1   2 2 1   1   nan nan nan nan nan
        nan nan nan 1   1   2   2   2 2 2   2   1   1   nan nan nan
        nan 1   1   2   2   2   2   2 2 2   2   2   2   1   1   nan
        1   1   1   1   1   2   2   2 2 2   2   1   1   1   1   1
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
```

```
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   1   1   1 1 1   1   1   nan nan nan nan ];
  set(fig,'pointershapecdata', pd,...
          'pointershapehotspot', [ 1 8 ] , ...
          'pointer','custom');
    case 'SOM leftright'
  pd = [ nan nan nan nan 1   nan nan nan nan nan nan 1  nan nan nan nan
        nan nan nan 1   1   nan nan nan nan nan nan 1  1   nan nan nan
        nan nan nan 1   1   nan nan nan nan nan nan 1  1   nan nan nan
        nan nan 1   2   1   nan nan nan nan nan nan 1  2   1   nan nan
        nan nan 1   2   1   1   1   1   1   1   1   1  2   1   nan nan
        nan 1   2   2   2   2   2   2   2   2   2   2  2   2   1   nan
        nan 1   2   2   2   2   2   2   2   2   2   2  2   2   1   nan
        1   2   2   2   2   2   2   2   2   2   2   2  2   2   2   1
        1   2   2   2   2   2   2   2   2   2   2   2  2   2   2   1
        nan 1   2   2   2   2   2   2   2   2   2   2  2   2   1   nan
        nan 1   2   2   2   2   2   2   2   2   2   2  2   2   1   nan
        nan nan 1   2   1   1   1   1   1   1   1   1  2   1   nan nan
        nan nan 1   2   1   nan nan nan nan nan nan 1  2   1   nan nan
        nan nan nan 1   1   nan nan nan nan nan nan 1  1   nan nan nan
        nan nan nan 1   1   nan nan nan nan nan nan 1  1   nan nan nan
        nan nan nan nan 1  nan nan nan nan nan nan 1  nan nan nan nan ];
  set(fig,'pointershapecdata', pd,...
          'pointershapehotspot', [ 8 8 ] , ...
          'pointer','custom');
    case 'SOM topbottom'
  pd = [ nan nan nan nan nan nan nan 1 1 nan nan nan nan nan nan nan
        nan nan nan nan nan 1   1   2 2 1   1   nan nan nan nan nan
        nan nan nan 1   1   2   2   2 2 2   2   1   1   nan nan nan
        nan 1   1   2   2   2   2   2 2 2   2   2   2   1   1   nan
        1   1   1   1   1   2   2   2 2 2   2   1   1   1   1   1
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        nan nan nan nan 1   2   2   2 2 2   2   1   nan nan nan nan
        1   1   1   1   1   2   2   2 2 2   2   1   1   1   1   1
        nan 1   1   2   2   2   2   2 2 2   2   2   2   1   1   nan
        nan nan nan 1   1   2   2   2 2 2   2   1   1   nan nan nan
        nan nan nan nan nan 1   1   2 2 1   1   nan nan nan nan nan
        nan nan nan nan nan nan nan 1 1 nan nan nan nan nan nan nan ];
  set(fig,'pointershapecdata', pd,...
          'pointershapehotspot', [ 8 8 ] , ...
          'pointer','custom');
     otherwise
  % Set it to the string passed in
   set(fig,'pointer', ptr);
end
```

上面程序代码的功能是设置不同情况下光标指针的形状，通过设置“PointerShapeData”属性，程序代码设置了不同的指针形状。在MATLAB中，“PointerShapeData”属性表示定义了16 × 16个像素组成的光标指针形状，该矩阵的元素只能选择1、2和NAN三种数值。其中，数值1代表的是黑色，数值2代表的是白色，NAN表示的是透明颜色。最后，将上面的程序代码保存为“setpointer.m”文件。

step 2 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function showarrowtip (arrow)
% Display a tip for ARROW.
% Depends on tipdata being set on the handle to ARROW.
  d=getappdata(gcf,'sliceomatic');
    if isempty(arrow)
    tipdata = [];
  else
        ctrlarrow = getappdata(arrow,'controlarrow');
        if ~isempty(ctrlarrow)
      arrow = ctrlarrow(2);
    end
        tipdata = getappdata(arrow,'tipdata');
  end
% 显示光标提示的文字信息
    if ~isempty(tipdata)
      set(d.tip,'parent',tipdata.parentaxes, ...
              'string',sprintf('Value: %1.3f',tipdata.value),...
              'units','data', ...
              'position', tipdata.position, ...
              'verticalalignment', tipdata.verticalalign,...
              'horizontalalignment', tipdata.horizontalalign);
    set(d.tip,'units','pixels');
    set(d.tip,'visible','on');
      else
        set(d.tip,'visible','off');
      end
```

上面程序代码的功能是显示光标指针的尖端部分，之所以编写上面的程序代码，是为了当用户使用鼠标选择相应的对象时，显示光标提示内容。最后，将上面的程序代码保存为“showarrowtip.m”文件。

step 2 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function [a, s]=getarrowslice
% Return the Arrow and Slice based on the GCO
  if isempty(getappdata(gco,'controlarrow')) && ...
        isempty(getappdata(gco,'isosurface'))
    a = gco;
    s = getappdata(a,'arrowslice');
    if isempty(s)
      s=getappdata(a,'arrowiso');
    end
  else
    s = gco;
```

```
    if ~isempty(getappdata(s,'isosurface'))
      s=getappdata(s,'isosurface');
    end
    a = getappdata(s,'controlarrow');
  end
```

上面程序代码的主要功能是，返回当前图形对象的箭头对象和切面对象，之所以编写上面的程序代码，其目的在于根据当前图形中的箭头对象信息进行操作。最后，将上面的程序代码保存为 "getarrowslice.m" 文件。



处理对象的属性

延续上面小节的步骤。

step 1 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的 "File" ⇨ "New" ⇨ "M-file" 命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function popset(handle,prop)
%选取句柄对象的属性域名列表
  proplist=fieldnames(get(handle(1)));
  prop=proplist{ strcmpi(prop,proplist)} ;
  appstr = [ prop '_hgstack'];
  for k=1:prod(size(handle))
         olds = getappdata(handle(k),appstr);
    if length(olds) <= 1
      error([ 'Nothing left to pop for property ' prop '.']);
    end
    set(handle(k),prop,olds{ 1} );
    setappdata(handle(k),appstr,olds{ 2:end} );
  end
```

上面程序代码的主要功能是，显示某个数据组中的对象属性数值。最后，将上面的程序代码保存为 "popset.m" 文件。

step 2 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的 "File" ⇨ "New" ⇨ "M-file" 命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function pushset(handle,prop,value)
%选取句柄对象的属性域名列表
  proplist=fieldnames(get(handle(1)));
  prop=proplist{ strcmpi(prop,proplist)} ;
  appstr = [ prop '_hgstack'];
  for k=1:prod(size(handle))
    oldv = get(handle(k),prop);
olds = getappdata(handle(k),appstr);
%设置句柄对象的属性值
    set(handle(k),prop,value);
    setappdata(handle(k),appstr,{ oldv olds });
  end
```

上面程序代码的主要功能是，设置新的对象属性数值。为了方便用户在后面程序中调用该代码，将上面的程序代码保存为 "pushset.m" 文件。

step 1 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的 "File" ⇨ "New" ⇨ "M-file" 命

令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function tf=propcheck(obj, prop, value)
% 检测某对象是否包含具体的属性值
  try
    v=get(obj,prop);
  catch
    tf = 0;
    return
  end
  if isa(v,class(value))
    if isa(v,'char')
      tf=strcmp(v,value);
    else
      if v==value
        tf=1;
      else
        tf=0;
      end
    end
  else
    tf=0;
  end
```

上面程序代码的主要功能是，检查图形对象是否包含某项属性值，程序代码的结构并不复杂，建议用户自行理解。最后，将该程序代码保存为"propcheck.m"文件。

step 4 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
ffunction slowset(handle, prop, value, increment)
%设置对象的属性值
  global INCREMENT;
    if nargin == 4
    INCREMENT = increment;
    if INCREMENT==0
      INCREMENT=1;
    end
  else
    INCREMENT=10;
  end
  % 获取句柄对象的属性域名类标
    proplist=fieldnames(get(handle(1)));
  tprop={ proplist{ strncmpi(prop,proplist,length(prop))} };
  prop=tprop{ 1} ;
    hp = [];
    for i = 1:length(handle)
    hp(i).handle = handle(i);
    hp(i).start = get(hp(i).handle,prop);
    hp(i).end = value;
    if isnumeric(hp(i).end) && isnumeric(hp(i).start)
      hp(i).values = VectorCalc(hp(i));
else
%设置对应的属性
      set(hp(i).handle,prop,value);
```

```
      hp(i).values = [];
    end
  end
    for inc = 1:INCREMENT
    for i = 1:length(handle)
      if ~isempty(hp(i).values)
        newval = reshape(hp(i).values(inc,:,:,:),...
                         size(hp(i).start,1),...
                         size(hp(i).start,2));
% 设置对应的属性
              set(hp(i).handle,prop,newval);
      end
    end
    pause(.05)
  end
  function values =  VectorCalc(hp)
  global INCREMENT;
    s = prod(size(hp.end));
    values = ones(INCREMENT,size(hp.end,1), size(hp.end,2),size(hp.end,3));
    for c = 1:s
    newval = linspace(hp.start(c),hp.end(c),INCREMENT);
    values(:,c) = newval';
  end
    values = reshape(values, INCREMENT, size(hp.end,1), size(hp.end,2), ...
                  size(hp.end,3));
```

上面程序代码的主要功能是，设置图形对象的属性值，和MATLAB内置的set函数功能类似，一次只能为某个图形对象设置单个属性值。最后，将上面的程序代码保存为"slowset.m"文件。

step 5 单击命令窗口工具栏中的 □ 按钮，或者选择编辑栏中的"File" ⇨ "New" ⇨ "M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function setvolumerange
% Query for a new volume range based on the sliceomatic gui
% which should be GCF
   %d=getappdata(fig,'sliceomatic');
    p=get(fig,'position');
  np=[ p(1)+20 p(2)+30 400 200];
% 创建新的图形窗口
  figure('position',np);
% 设置图形窗口的控件
  uicontrol('units','norm','style','text','string','X Range',...
            'position',[ 0 .6 .3 .3]);
  uicontrol('units','norm','style','text','string','Y Range',...
            'position',[ 0 .3 .3 .3]);
  uicontrol('units','norm','style','text','string','Z Range',...
            'position',[ 0 0  .3 .3]);
```

上面程序代码的功能是，重新设置三维图形数据的数值范围。当用户运行上面的程序代码时，MATLAB会显示一个新的图形界面，提示用户输入新的图形位置参数。

编写主程序代码

延续上面小节的步骤。

step 1 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的 "File" ⇨ "New" ⇨ "M-file" 命令，打开一个空白的 M 文件编辑器，然后在 M 文件编辑器中输入下面的代码：

```
function p=arrow(parent,dir,pos)
%根据光标不同的方向设置不同的光标形状数组
 switch dir
   case 'down'
    pts=[ 0 1; -2 3; -1 3; -1 5; 1 5; 1 3; 2 3 ];
    mp = 'SOM leftright';
   case 'up'
    pts=[ 0 5; 2 3; 1 3; 1 1; -1 1; -1 3; -2 3; ];
    mp = 'SOM leftright';
   case 'right'
    pts=[ 5 0; 3 -2; 3 -1; 1 -1; 1 1; 3 1; 3 2 ];
    mp = 'SOM topbottom';
   case 'left'
    pts=[ 1 0; 3 2; 3 1; 5 1; 5 -1; 3 -1; 3 -2 ];
    mp = 'SOM topbottom';
  end
  f=[1 2 7; 3 4 5; 3 5 6 ];
 %处理光标的外观属性
  if pos(1)
    lim=get(parent,'xlim');
    fivep=abs(lim(1)-lim(2))/15/5;
    pts(:,1)=pts(:,1)*fivep+pos(1);
  elseif pos(2)
    lim=get(parent,'ylim');
    fivep=abs(lim(1)-lim(2))/15/5;
    pts(:,2)=pts(:,2)*fivep+pos(2);
  end
  % 创建patch对象
 p(1)=patch('vertices',pts,'faces',1:size(pts,1),'facec','n','edgec','k',...
            'linewidth',2,'hittest','off',...
            'parent',parent);
 p(2)=patch('vertices',pts,'faces',f,'facec','g','facea',.5,'edgec','n',...
            'parent',parent,'tag','sliceomaticarrow');
 %向patch对象中添加数据
  setappdata(p(2),'arrowcenter',pos);
  setappdata(p(2),'arrowedge',p(1));
  setappdata(p(2),'motionpointer',mp);
```

在前面的小节中，曾经专门编写过处理箭头指针对象的代码，但是本实例中还需要对箭头进行其他的处理。在上面的程序代码中，首先根据箭头指针的移动方向来显示不同的箭头形状，然后创建块对象，将箭头对象接触过的图形界面位置数据保存到块对象中。

step 2 在上面步骤打开的 M 文件编辑器中，输入下面的代码：

```
function movetipforarrow(arrow, ax, value, position, va, ha)
% 显示Slice光标的数值提示内容，显示控件的数值
  tipdata.parentaxes = ax;
  tipdata.value = value;
```

```
  tipdata.position = position;
  tipdata.verticalalign = va;
  tipdata.horizontalalign = ha;
   setappdata(arrow, 'tipdata', tipdata);
  showarrowtip(arrow);
```

上面程序代码的功能是，为某个切面箭头创建当前设置，同时显示对应切面的控件数值。

step 3 在上面步骤打开的 M 文件编辑器中，输入下面的代码：

```
function localcontour(slice,oldcontour,levels)
% 在切片图上绘制等高线
% 程序代码中没有引用 CONTOURSLICE 命令，而是使用特别的切片图
  d=getappdata(gcf,'sliceomatic');
cdata = get(slice,'cdata');
%获取切片图的类型信息
  st = getappdata(slice,'slicetype');
  % 计算新的等高线数值
  if nargin < 3
    if isnan(d.zmesh)==1
      c = contourc(cdata);
else
%根据切面类型的不同，绘制不同的等高线
      switch st
       case 'X'
        c = contours(d.zmesh,d.ymesh,cdata);
       case 'Y'
        c = contours(d.zmesh,d.xmesh,cdata);
       case'Z'
        c = contours(d.xmesh,d.ymesh,cdata);
      end
    end
  else
%在绘制等高线的时候，设置 levels 参数的数值
    if isnan(d.zmesh)==1
      c = contourc(cdata,levels);
    else
      switch st
       case 'X'
        c = contours(d.zmesh,d.ymesh,cdata,levels);
       case 'Y'
        c = contours(d.zmesh,d.xmesh,cdata,levels);
       case 'Z'
        c = contours(d.xmesh,d.ymesh,cdata,levels);
      end
    end
  end
  newvertices = [];
  newfaces = {};
  longest = 1;
  cdata = [];
  limit = size(c,2);
  i = 1;
  while(i < limit)
    z_level = c(1,i);
    npoints = c(2,i);
```

```
    nexti = i+npoints+1;
    xdata = c(1,i+1:i+npoints);
ydata = c(2,i+1:i+npoints);
%根据选择的不同截面类型，计算参数vertices的数值
    switch st
     case 'X'
      xv = get(slice,'xdata');
      lzdata = xv(1,1) + 0*xdata;
      vertices = [ lzdata.', ydata.', xdata.'];
     case 'Y'
      yv = get(slice,'ydata');
      lzdata = yv(1,1) + 0*xdata;
      vertices = [ ydata.', lzdata.', xdata.'];
     case 'Z'
      zv = get(slice,'zdata');
      lzdata = zv(1,1) + 0*xdata;
      vertices = [ xdata.', ydata.', lzdata.'];
    end
    faces = 1:length(vertices);
    faces = faces + size(newvertices,1);
    longest=max(longest,size(faces,2));
    newvertices = [ newvertices ; vertices ];
    newfaces{ end+1} = faces;
    tcdata =  (z_level + 0*xdata).';
    cdata = [ cdata; tcdata ]; % need to be same size as faces
        i = nexti;
  end
 %添加nan参数，结束循环
  newvertices = [ newvertices ; nan nan nan ];
  cdata = [ cdata ; nan ];
    vertmax = size(newvertices,1);
     faces = [];
  for i = 1:size(newfaces,2)
    faces = [ faces;
              newfaces{ i}  ones(1,longest-size(newfaces{ i} ,2))*vertmax
vertmax ];
  end
%设置等高线的属性
    if isempty(oldcontour)
      oldcontour  = patch('facecolor','none', 'edgecolor',d.
defcontourcolor,...
                         'linewidth',d.defcontourlinewidth);
    try
      set(oldcontour,'linesmoothing',d.defcontoursmooth);
    catch
    end
    setappdata(slice,'contour',oldcontour);
  end
  set(oldcontour,'vertices',newvertices,...
                 'faces',faces,...
                 'facevertexcdata',cdata);
```

上面程序代码的功能是，在绘制的切面图上添加等高线。熟悉MATLAB的读者也许想知道，为何没有使用CONTOURSLICE的内置函数。这是因为，该程序代码处理的Slice截面并不通过使用MATLAB通用代码绘制的Slice截面。

step 4 在上面步骤打开的 M 文件编辑器中，输入下面的代码：

```
function p=localisosurface(volume, data, datanormals, value, oldiso)
% 处理 Isosurface 对象的程序代码
% 设置当前图形窗口的属性
  pushset(gcf, 'pointer','watch');
% 获取当前图形窗口的 sliceomatic' 截面的图形信息
  d=getappdata(gcf,'sliceomatic');
% 绘制 Isosurface 对象
  fv = isosurface(volume{ :} ,data, value);
  clim=get(gca,'clim');
% 设置图形的色图属性
  cmap=get(gcf,'colormap');
  clen=clim(2)-clim(1);
  idx=floor((value-clim(1))*length(cmap)/clen);
% 设置 Isosurface 对象的属性
 if nargin==5
    try
      set(oldiso,fv,'facecolor',cmap(idx,:));
    catch
      set(oldiso,fv,'facecolor','none');
    end
    p=oldiso;
    cap=getappdata(p,'isosurfacecap');
    if ~isempty(cap)
      localisocaps(p,cap);
    end
  else
% 绘制 patch 对象
    if isnan(d.xmesh)==1
       p=patch(fv,'edgecolor','none','facecolor',cmap(idx,:), 'tag',
'sliceomaticisosurface');
    else
       p=patch(fv,'edgecolor','none','facecolor',cmap(idx,:),'tag',
'sliceomaticisosurface');
    end
    % d=getappdata(gcf,'sliceomatic');
    % 设置图形的光照属性
    switch d.deflight
     case 'flat'
      set(p,'facelighting','flat');
     case 'smooth'
      set(p,'facelighting','phong');
    end
    setappdata(p,'isosurfacecap',[]);
  end
  setappdata(p,'isosurfacevalue',value);
  setappdata(p,'isosurfacedata',data);
  reducepatch(p,10000);
  isonormals(volume{ :} ,datanormals,p);
% 设置图形的属性
  popset(gcf,'pointer');
```

上面程序代码的功能是处理 Isosurface 的各种属性，具体分析如下。

- 在 MATLAB 中，Isosurface 表示的是在某数值体积内每个顶点所在地方数值相等的表面。该曲面和等高线有些类似，因为这两种曲面都是表示数值相等的位置。
- 当需要确定某个三维空间内取得某临界值（threshold value）的数值范围，或者需要判断某三维空间内数据分布情况时，Isosurface 曲面是相当有用的，需要提醒读者注意的是，该三维数据空间必须是有限的。
- 在 MATLAB 中，创建 Isosurface 曲面的常见命令是 isosurface 和 patch 命令，而求解 Isosurface 曲面法线的命令是 isonormals。因此，在上面的程序代码中，首先使用 isosurface 命令创建该 Isosurface 曲面，然后设置曲面的表面颜色（facecolor）和边缘颜色（edgecolor）属性，最后使用 isonormals 创建 Isosurface 曲面的法线。



为了减少整个程序代码的运行时间，在上面的程序代码中使用了reducepatch命令将创建的"块"对象的表面数目减少到1000个。关于该命令的详细使用方法，请读者参阅 MATLAB 的帮助文件。

step 5 在上面步骤打开的 M 文件编辑器中，输入下面的代码：

```
function p=localisocaps(isosurface,isocap)
% 处理 Isocap 对象
  if nargin<2 || ~strcmp(get(isocap,'visible'),'off')
    d=getappdata(gcf,'sliceomatic');
    data=getappdata(isosurface,'isosurfacedata');
    if isnan(d.xmesh)==1
      caps=isocaps(data,getappdata(isosurface,'isosurfacevalue'));
    else
      caps=isocaps(d.xmesh,d.ymesh,d.zmesh,data,getappdata(isosurface,
'isosurfacevalue'));
    end
  end
  if nargin==2
    if ~strcmp(get(isocap,'visible'),'off')
      set(isocap,caps);
    end
    p=isocap;
  else
% 绘制 isocap 对象
    p=patch(caps,'edgecolor','none','facecolor','flat', 'facelighting',
              'none','tag','sliceomaticisocap');
    setappdata(p,'isosurface',isosurface);
    setappdata(isosurface,'isosurfacecap',p);
d=getappdata(gcf,'sliceomatic');
    % 设置图像的光照属性
    switch d.defcolor
     case 'faceted'
      set(p,'facec','flat','edgec','black');
     case 'flat'
      set(p,'facec','flat','edgec','none');
     case 'interp'
      set(p,'facec','interp','edgec','none');
     case 'texture'
      set(p,'facec','flat','edgec','none');
     case 'none'
```

```
        set(p,'facec','none','edgec','none');
    end
    %设置图形的透明属性
        switch d.defalpha
         case 'none'
          set(p,'facea',1);
         case 'flat'
          set(p,'facea','flat');
         case 'interp'
          set(p,'facea','interp');
         case 'texture'
          set(p,'facea','flat');
        end
      end
```

上面程序代码的功能是处理Isocap的各种属性。在MATLAB中，Isocap表示的是在Isosurface曲面的空间限制平面，这些平面就组成了Isosurface曲面的空间范围。Isocap提供了Isosurface曲面内部空间的横截面视角。



在MATLAB中，通常使用isocaps命令来创建Isocap的数据，然后将该数据传递给patch命令，创建出Isocap对象。和Isosurface曲面类似，Isocap对象在MATLAB中都属于patch对象，用户可以对其设置所有patch相关的属性。

step 6 在上面步骤打开的M文件编辑器中，输入下面的代码：

```
function s=localslice(data, X, Y, Z, oldslice)
%处理切面对象
  s=[];
  d=getappdata(gcf,'sliceomatic');
  ds=size(data);
  %如果用户选取的是x向切面
  if ~isempty(X)
  %处理z向数据
    xi=round(X);
    if isnan(d.xmesh) == 1
      if xi > 0 && xi <= ds(2)
%根据需要重新获取颜色信息数组
        cdata=reshape(data(:,xi,:),ds(1),ds(3));
%产生绘制图形的数据格点
        [xdata ydata zdata]=meshgrid(xi,1:ds(1),1:ds(3));
%返回切面类型信息
        st = 'X';
      else
        return
      end
   else
   %如果x坐标轴的方向是反向的
      if isequal(d.xdir,'reverse')==1
%将x向数据矩阵进行转置
        locate_xi=histc(xi,flipdim(d.xmesh,2));
        slice_number=find(locate_xi);
        slice_number=length(d.xmesh)-slice_number+1;
      else
```

```
            locate_xi=histc(xi,d.xmesh);
            slice_number=find(locate_xi);
          end
          if ~isempty(slice_number) && slice_number > 0 && slice_number <= ds(2)
%根据需要重新获取颜色信息数组
            cdata=reshape(data(:,slice_number,:),ds(1),ds(3));
%产生绘制图形的数据格点
            [xdata ydata zdata]=meshgrid(X,d.ymesh,d.zmesh);
%返回切面类型信息
            st = 'X';
          else
            return
          end
      end
      %如果用户选取Y向切面
      elseif ~isempty(Y)
      %处理Y向数据
       yi=round(Y);
       if isnan(d.ymesh) == 1
         if yi > 0 && yi <= ds(1)
%根据需要重新获取颜色信息数组
            cdata=reshape(data(yi,:,:),ds(2),ds(3));
%产生绘制图形的数据格点
            [xdata ydata zdata]=meshgrid(1:ds(2),yi,1:ds(3));
%返回切面类型信息
            st = 'Y';
         else
            return
         end
      else
      %如果y坐标轴的方向是反向的
         if isequal(d.ydir,'reverse')==1
%将x向数据矩阵进行转置
            locate_yi=histc(yi,flipdim(d.ymesh,2));
            slice_number=find(locate_yi);
            slice_number=length(d.ymesh)-slice_number+1;
         else
            locate_yi=histc(yi,d.ymesh);
            slice_number=find(locate_yi);
         end
         if ~isempty(slice_number) && slice_number > 0 && slice_number <= ds(1)
%根据需要重新获取颜色信息数组
            cdata=reshape(data(slice_number,:,:),ds(2),ds(3));
%产生绘制图形的数据格点
            [xdata ydata zdata]=meshgrid(d.xmesh,Y,d.zmesh);
%返回切面类型信息
            st = 'Y';
         else
            return
         end
       end
       %如果用户选取的是Z向切面
       elseif ~isempty(Z)
       %处理Z向数据
       zi=round(Z);
       if isnan(d.zmesh) == 1
```

```
      if zi > 0 && zi <= ds(3)
% 根据需要重新获取颜色信息数组
        cdata=reshape(data(:,:,zi),ds(1),ds(2));
% 产生绘制切面的数据格点
        [ xdata ydata zdata] =meshgrid(1:ds(2),1:ds(1),zi);
% 返回切面类型信息
        st = 'Z';
      else
        return
      end
    else
    % 如果 z 坐标轴的方向是反向的
      if isequal(d.zdir,'reverse')==1
% 将 z 向数据矩阵进行转置
        locate_zi=histc(zi,flipdim(d.zmesh,2));
        slice_number=find(locate_zi);
        slice_number=length(d.zmesh)-slice_number+1;
      else
        locate_zi=histc(zi,d.zmesh);
        slice_number=find(locate_zi);
      end
      if ~isempty(slice_number) && slice_number > 0 && slice_number <= ds(3)
% 根据需要重新获取颜色信息数组
        cdata=reshape(data(:,:,slice_number),ds(1),ds(2));
% 产生绘制切面的数据格点
        [ xdata ydata zdata] =meshgrid(d.xmesh,d.ymesh,Z);
% 返回切面类型信息
        st = 'Z';
      else
        return
      end
    end
  else
    error('Nothing was passed into LOCALSLICE.');
  end
% 减少数据数组中的独维
  cdata=squeeze(cdata);
  xdata=squeeze(xdata);
  ydata=squeeze(ydata);
  zdata=squeeze(zdata);
  if nargin == 5
    % 设置原来切片面的属性
    set(oldslice,'cdata',cdata,'alphadata',cdata, 'xdata',xdata, ...
                'ydata',ydata, 'zdata',zdata);
    s=oldslice;
    % 检测图形表面的属性
    if propcheck(s,'facec','texturemap')
      textureizeslice(s,'on');
    end
    setappdata(s,'slicetype',st);
  else
    % 绘图表面图
    news=surface('cdata',cdata,'alphadata',cdata, 'xdata',xdata, ...
                'ydata',ydata, 'zdata',zdata);
     set(news,'alphadata',cdata,'alphadatamapping','scaled','tag',
'sliceomaticslice',...
```

```
                'facelighting','none',...
                'uicontextmenu',d.uic);
s=news;
% 设置截面类型
setappdata(s,'slicetype',st);
% 设置图像的照明属性
    switch d.defcolor
     case 'faceted'
      set(s,'facec','flat','edgec','k');
     case 'flat'
      set(s,'facec','flat','edgec','n');
     case 'interp'
      set(s,'facec','interp','edgec','n');
     case 'texture'
      set(s,'facec','texture','edgec','n');
end
% 设置图形的透明属性
    switch d.defalpha
     case 'none'
      set(s,'facea',1);
     case 'flat'
      set(s,'facea','flat');
     case 'interp'
      set(s,'facea','interp');
     case 'texture'
      set(s,'facea','texture');
    end
  end
% 获取图形中等高线的数值信息
 contour = getappdata(s,'contour')
% 设置等高线的属性
  if ~isempty(contour)
    try
      levels = getappdata(s, 'contourlevels');
      if isempty(levels)~=1
        localcontour(s,contour,levels);
      else
        localcontour(s, contour);
      end
    catch
      localcontour(s, contour);
    end
  end
```

在上面的程序代码中，首先判断Slice截面的数目，然后将截面数据转换为二维数据，最后，使用surface命令来创建表面对象，并设置该表面对象的各种属性，主要设置图形对象的照明属性和透明属性，具体的程序代码就不详细解释了，请读者自行分析和运行。

step 7 在上面步骤打开的 M 文件编辑器中，输入下面的代码：

```
function textureizeslice(slice,onoff)
% 实现一般的切面图和材质化的切面图之间的相互转换
    for k=1:prod(size(slice))
    d=getappdata(slice(k),'textureoptimizeations');
```

```
    switch onoff
     case 'on'
%获取切面各向的数据类型
       d.xdata=get(slice(k),'xdata');
       d.ydata=get(slice(k),'ydata');
       d.zdata=get(slice(k),'zdata');
%设置材质化图形的数据类型
       setappdata(slice(k),'textureoptimizeations',d);
%返回符合材质贴图的数据数组
       if max(size(d.xdata)==1)
         nx=[ d.xdata(1) d.xdata(end)];
       else
         nx=[ d.xdata(1,1)   d.xdata(1,end);
             d.xdata(end,1) d.xdata(end,end)];
       end
       if max(size(d.ydata)==1)
         ny=[ d.ydata(1) d.ydata(end)];
       else
         ny=[ d.ydata(1,1)   d.ydata(1,end);
             d.ydata(end,1) d.ydata(end,end)];
       end
       if max(size(d.zdata)==1)
         nz=[ d.zdata(1) d.zdata(end)];
       else
         nz=[ d.zdata(1,1)   d.zdata(1,end);
             d.zdata(end,1) d.zdata(end,end)];
       end
%设置图形的材质化属性
       set(slice(k),'xdata',nx, 'ydata', ny, 'zdata', nz,...
                    'facec','texturemap');
       if ischar(get(slice(k),'facea'))
         set(slice(k),'facea','texturemap');
       end
       if ischar(get(slice(k),'facec'))
         set(slice(k),'facec','texturemap');
       end
      case 'off'
       if ~isempty(d)
         set(slice(k),'xdata',d.xdata,'ydata',d.ydata,'zdata',d.zdata);
         setappdata(slice(k),'textureoptimizeations',[]);
       end
%将材质化的图形转换为一般图形
      if ischar(get(slice(k),'facea')) && strcmp(get(slice(k),'facea'),
'texturemap')
         set(slice(k),'facea','flat');
       end
      if ischar(get(slice(k),'facec')) && strcmp(get(slice(k),'facec'),
'texturemap')
         set(slice(k),'facec','flat');
       end
    end
  end
```

上面的程序代码的主要功能是将正常的切片图和质地处理过的切片图进行转换。在MATLAB中，材质贴图（Texture mapping）是将一个二维图像映射到三维图像的重要手段，其程序

代码将颜色数据进行转换，一次可以转换到平面图形中。



在这些图形转换方法中，用户可以选择某种图形材质，例如森林、谷类等，将这些材质添加到表面对象中，而无须进行任何的几何运算。关于该材质命令的具体方法请读者查阅相应的帮助文件。

step 8 在上面步骤打开的 M 文件编辑器中，输入下面的代码：

```
function ss=allSlices
  ss=findobj(gcf,'type','surface','tag','sliceomaticslice');

function ss=allIsos
  ss=findobj(gcf,'type','patch','tag','sliceomaticisosurface');

function ss=allCaps
  ss=findobj(gcf,'type','patch','tag','sliceomaticisocap');
```

上面的程序中使用findobj命令来返回所有标签为sliceomaticslice、sliceomaticisosurface和sliceomaticisocap的"块"对象的句柄。

step 9 在上面步骤打开的 M 文件编辑器中，输入下面的代码：

```
function dragprep(arrowtodrag)
  arrows=findall(gcf,'tag','sliceomaticarrow');
% 设置图形中的箭头属性：表明颜色、透明度等
  pushset(arrows,'facecolor',[1 0 0]);
  pushset(arrows,'facealpha',.2);
  pushset(arrowtodrag,'facecolor',[0 1 0]);
  pushset(arrowtodrag,'facealpha',.7);
% 返回切面信息
  slices=allSlices;
  for i=1:length(slices)
    fa=get(slices(i),'facea');
    % 设置所有切面的透明和边缘颜色属性
    if isa(fa,'double') && fa>.3
      pushset(slices(i),'facealpha',.3);
      pushset(slices(i),'edgecolor','n');
    else
      pushset(slices(i),'facealpha',fa);
      pushset(slices(i),'edgecolor',get(slices(i),'edgec'));
    end
  end
% 返回所有的isosurface对象
  isosurfs=allIsos;
  for i=1:length(isosurfs)
    % 设置所有isosurface对象的透明和边缘颜色属性
    fa=get(isosurfs(i),'facea');
    if isa(fa,'double') && fa>.3
      pushset(isosurfs(i),'facealpha',.3);
      pushset(isosurfs(i),'edgecolor','n');
    else
      pushset(isosurfs(i),'facealpha',fa);
      pushset(isosurfs(i),'edgecolor',get(isosurfs(i),'edgec'));
```

```
    end
    %设置 cap 对象的属性
    cap=getappdata(isosurfs(i),'isosurfacecap');
    if ~isempty(cap)
      pushset(cap,'visible','off');
    end
  end
  ss=getappdata(arrowtodrag,'arrowslice');
  if isempty(ss)
    ss=getappdata(arrowtodrag,'arrowiso');
  end
%设置用户运行箭头的属性
  popset(ss,'facealpha');
  popset(ss,'edgecolor');
  pushset(gcf,'windowbuttonupfcn','sliceomatic up');
  pushset(gcf,'windowbuttonmotionfcn','sliceomatic motion');
  % Doing this makes the tip invisible when visible is on.
  showarrowtip(arrowtodrag);
```

在上面的程序代码中，首先使用程序语句设置光标的属性，然后通过上面步骤返回的对象句柄设置图形对象的属性。上面的程序代码并不复杂，请读者自行分析。

step 10 在上面步骤打开的 M 文件编辑器中，输入下面的代码：

```
function dragfinis(arrowtodrag)
  arrows=findall(gcf,'tag','sliceomaticarrow');
%设置箭头对象的属性
  popset(arrowtodrag,'facecolor');
  popset(arrowtodrag,'facealpha');
  popset(arrows,'facecolor');
  popset(arrows,'facealpha');
  ss=getappdata(arrowtodrag,'arrowslice');
  if isempty(ss)
    ss=getappdata(arrowtodrag,'arrowiso');
  end
  % These pushes are junk which will be undone when all slices or
  % isosurfs are reset below.
  pushset(ss,'facealpha',1);
  pushset(ss,'edgecolor','k');
  slices=allSlices;
%设置切面属性
  if ~isempty(slices)
    popset(slices,'facealpha');
    popset(slices,'edgecolor');
  end
  isosurfs=allIsos;
%设置 isosurface 对象的属性
  if ~isempty(isosurfs)
    popset(isosurfs,'facealpha');
    popset(isosurfs,'edgecolor');
  end
  d=getappdata(gcf,'sliceomatic');
    if isnan(d.xmesh)==1
    for i=1:length(isosurfs)
      cap=getappdata(isosurfs(i),'isosurfacecap');
      if ~isempty(cap)
```

```
        popset(cap,'visible');
        localisocaps(isosurfs(i),cap);
      end
      if getappdata(isosurfs(i), 'reduced')
        setappdata(isosurfs(i),'reduced',0);
        localisosurface({},d.data,d.smooth,...
                        getappdata(isosurfs(i),'isosurfacevalue'),...
                        isosurfs(i));
      end
    end
  else
for i=1:length(isosurfs)
% 设置 cap 对象的属性
      cap=getappdata(isosurfs(i),'isosurfacecap');
      if ~isempty(cap)
        popset(cap,'visible');
        localisocaps(isosurfs(i),cap);
      end
      if getappdata(isosurfs(i), 'reduced')
        setappdata(isosurfs(i),'reduced',0);
        realvolume={ d.xmesh d.ymesh d.zmesh};
        localisosurface(realvolume,d.data,d.smooth,...
                        getappdata(isosurfs(i),'isosurfaoevalue'),...
                        isosurfs(i));
      end
    end
  end
% 设置当前图形的调用函数
  popset(gcf,'windowbuttonupfcn');
  popset(gcf,'windowbuttonmotionfcn');
  showarrowtip([]);
    buf = get(gcf,'windowbuttonupfcn');
  if ~strcmp(buf,'')
    eval(buf);
  end
```

在上面的程序代码中，首先设置了箭头光标的属性，然后将光标分为Slice截面选取光标和ISO颜色系统选取光标来分别设置属性对象。

step 11 在上面步骤打开的M文件编辑器中，输入下面的代码：

```
function sliceomatic(p1,p2,xmesh,ymesh,zmesh)
% SLICEOMATIC - Slice and isosurface volume exploration GUI
%
% SLICEOMATIC(DATA) - Use 3D double matrix DATA as a volume data
%
% 示例
%        x = -2:.2:2; y = -2:.25:2; z = -2:.16:2;
%        [X,Y,Z,] = meshgrid(x,y,z);
%        v = X .* exp(-X.^2 - Y.^2 - Z.^2);
%        sliceomatic(v,x,y,z)
%
  if nargin==0
% 当用户没有输入任何参数时，则使用下面的数据系列创建图形
    [x,y,z] = meshgrid(-2:.2:2, -2:.25:2, -2:.16:2);
    v = x .* exp(-x.^2 - y.^2 - z.^2);
```

```
      sliceomatic(v)
      return
    end
% 处理输入参数
    if isa(p1,'double')
          d.data=p1;
          if nargin>=4
          if nargin==4
          zmesh=ymesh;
          ymesh=xmesh;
          xmesh=p2;
        end
% 调用 sliceomaticfigure 函数绘制图形
        d = sliceomaticfigure(d,xmesh,ymesh,zmesh);
% 调用 sliceomaticsetdata 函数获取图形数据
        d = sliceomaticsetdata(d,xmesh,ymesh,zmesh);
      else
        d = sliceomaticfigure(d);
        d = sliceomaticsetdata(d);
      end
          setappdata(gcf,'sliceomatic',d);
    elseif isa(p1,'char')
      % Interpret commands
      d=getappdata(gcf,'sliceomatic');
      try
        switch p1
% 当参数 p1 的数值为 xnew 时
          case 'Xnew'
           if strcmp(get(gcf,'selectiontype'),'normal')
% 获取当前数据点的坐标
             pt=get(gcbo,'currentpoint');
% 获取当前图形的坐标轴属性
             axis(gcbo);
             X=pt(1,1);
% 创建箭头对象
             newa=arrow(gcbo,'down',[ X 0] );
% 设置坐标轴对象的属性
             set(gof,'currentaxes',d.axmain);
% 调用 localslice 函数绘制切面对象
             new=localslice(d.data, X, [] , [] );
% 设置控件箭头和切面箭头的属性数据
             setappdata(new,'controlarrow',newa);
             setappdata(newa(2),'arrowslice',new);
% 设置创建的切面对象属性
              set(new,'alphadata',get(new,'cdata'),'alphadatamapping',
'scaled');
% 设置箭头的属性
             set(newa,'buttondownfcn','sliceomatic Xmove');
% 设置现场菜单的属性
             set([ new newa] ,'uicontextmenu',d.uic);
             buf = get(gcf,'windowbuttonupfcn');
             if ~strcmp(buf,'')
               eval(buf);
             end
             d.draggedarrow=newa(2);
             dragprep(newa(2));
```

```
        setpointer(gcf,'SOM leftright');
        set(d.motionmetaslice,'visible','off');
      end
% 当参数 p1 的数值是 Ynew 时
% 参考前段程序代码中处理 xnew 的方法
    case 'Ynew'
      if strcmp(get(gcf,'selectiontype'),'normal')
        pt=get(gcbo,'currentpoint');
        Y=pt(1,2);
        newa=arrow(gcbo,'right',[ 0 Y] );
        set(gcf,'currentaxes',d.axmain);
        new=localslice(d.data, [] , Y, [] );
        setappdata(new,'controlarrow',newa);
        setappdata(newa(2),'arrowslice',new);
        set(new,'alphadata',get(new,'cdata'),'alphadatamapping','scaled');
        set(newa,'buttondownfcn','sliceomatic Ymove');
        set([ new newa] ,'uicontextmenu',d.uic);
        buf = get(gcf,'windowbuttonupfcn');
        if ~strcmp(buf,'')
          eval(buf);
        end
        d.draggedarrow=newa(2);
        dragprep(newa(2));
        setpointer(gcf,'SOM topbottom');
        set(d.motionmetaslice,'visible','off');
      end
% 当参数 p1 的数值是 znew 时
% 参考前段程序代码中处理 xnew 的方法
    case 'Znew'
      if strcmp(get(gcf,'selectiontype'),'normal')
        pt=get(gcbo,'currentpoint');
        Y=pt(1,2);
        newa=arrow(gcbo,'left', [ 0 Y] );
        set(gcf,'currentaxes',d.axmain);
        new=localslice(d.data, [] , [] , Y);
        set(new,'alphadata',get(new,'cdata'),'alphadatamapping','scaled');
        setappdata(new,'controlarrow',newa);
        setappdata(newa(2),'arrowslice',new);
        set(newa,'buttondownfcn','sliceomatic Zmove');
        set([ new newa] ,'uicontextmenu',d.uic);
        buf = get(gcf,'windowbuttonupfcn');
        if ~strcmp(buf,'')
          eval(buf);
        end
        d.draggedarrow=newa(2);
        dragprep(newa(2));
        setpointer(gcf,'SOM topbottom');
        set(d.motionmetaslice,'visible','off');
      end
% 当参数 p1 的数值为 ISO 时
% 参考前段程序代码中处理 xnew 的方法
    case 'ISO'
      if strcmp(get(gcf,'selectiontype'),'normal')
        pt=get(gcbo,'currentpoint');
        V=pt(1,1);
        newa=arrow(gcbo,'up',[ V 0] );
```

```
    set(gcf,'currentaxes',d.axmain);
    new=localisosurface(d.reducelims,d.reduce,d.reducesmooth,V);
    set([ newa new] ,'uicontextmenu',d.uiciso);
    setappdata(new,'controlarrow',newa);
    setappdata(new,'reduced',1);
    setappdata(newa(2),'arrowiso',new);
    set(newa,'buttondownfcn','sliceomatic ISOmove');
    buf = get(gcf,'windowbuttonupfcn');
    if ~strcmp(buf,'')
      eval(buf);
    end
    d.draggedarrow=newa(2);
    dragprep(newa(2));
    setpointer(gcf,'SOM leftright');
  end

 case 'Xmove'
  if strcmp(get(gcf,'selectiontype'),'normal')
    [ a s] =getarrowslice;
    d.draggedarrow=a;
    dragprep(a);
  end
 case 'Ymove'
  if strcmp(get(gcf,'selectiontype'),'normal')
    [ a s] =getarrowslice;
    d.draggedarrow=a;
    dragprep(a);
  end
 case 'Zmove'
  if strcmp(get(gcf,'selectiontype'),'normal')
    [ a s] =getarrowslice;
    d.draggedarrow=a;
    dragprep(a);
  end
 case 'ISOmove'
  if strcmp(get(gcf,'selectiontype'),'normal')
    [ a s] =getarrowslice;
    d.draggedarrow=a;
    dragprep(a);
  end
 case 'up'
  if strcmp(get(gcf,'selectiontype'),'normal')
    dragfinis(d.draggedarrow);
  end
 case 'motion'
  % Make sure our cursor is ok
  a=d.draggedarrow;                    % 绘制箭头
  s=getappdata(a,'arrowslice');        % The slice to 'move'
  if isempty(s)
    s=getappdata(a,'arrowiso');        % or the isosurface
  end
  aa=get(a,'parent');            % 箭头的当前坐标数值
  pos=getappdata(a,'arrowcenter');
  apos=get(aa,'currentpoint');

  % 设置坐标轴的数值属性
```

```
xlimits = get(aa,'xlim');
ylimits = get(aa,'ylim');
if apos(1,1) < xlimits(1)
  apos(1,1) = xlimits(1);
elseif apos(1,1) > xlimits(2)
  apos(1,1) = xlimits(2);
end
          if apos(1,2) < ylimits(1)
  apos(1,2) = ylimits(1);
elseif apos(1,2) > ylimits(2)
  apos(1,2) = ylimits(2);
end
  if aa==d.axx || aa==d.axiso
  % We are moving an X slice
  xdiff=apos(1,1)-pos(1,1);
  v=get(a,'vertices');
  v(:,1)=v(:,1)+xdiff;
  set([ a getappdata(a,'arrowedge')],'vertices',v);
  np=[  apos(1,1)  0 ];
  % This might be a slice, or an isosurface!
  if aa==d.axiso
   new=localisosurface(d.reducelims,d.reduce,d.reducesmooth,...
                      apos(1,1),s);
    setappdata(new,'reduced',1);
   movetipforarrow(a, aa, apos(1,1), [ apos(1,1) 6 ], 'bottom','center')
  else
    %disp([  'apos  =  ' num2str(apos(1,1))])
    %disp([  'pos   =  ' num2str(pos(1,1))])
    %disp([  'change=' num2str(round(apos(1,1))~=round(pos(1,1)))]);
    if round(apos(1,1))~=round(pos(1,1))
      localslice(d.data, apos(1,1), [], [],s);
    end
   movetipforarrow(a, aa, apos(1,1), [ apos(1,1) .5 ],'top','center')
  end
else
  % 当用户移动Y向切面或者Z向切面时
  ydiff=apos(1,2)-pos(1,2);
  v=get(a,'vertices');
  v(:,2)=v(:,2)+ydiff;
  set([ a getappdata(a,'arrowedge')],'vertices',v);
  np=[  0 apos(1,2) ];
  if aa==d.axy
    if round(apos(1,2))~=round(pos(1,2))
      localslice(d.data, [], apos(1,2), [], s);
    end
   movetipforarrow(a, aa, apos(1,2), [ 5.5 apos(1,2) ], 'middle','left');
  else
    if round(apos(1,2))~=round(pos(1,2))
      localslice(d.data, [], [], apos(1,2), s);
    end
   movetipforarrow(a, aa, apos(1,2), [ .5 apos(1,2) ], 'middle','right');
  end
end
setappdata(a,'arrowcenter',np);
try
  if isempty(get(gcf,'javaframe'))
```

```
          drawnow;
        end
      catch
        drawnow;
      end
    % IsoSurface 现场菜单子选项的回调函数
     case 'isotogglevisible'
      [ a s] =getarrowslice;
%修改 slice 切面的可视性属性
      if propcheck(s,'visible','on')
        set(s,'visible','off');
      else
        set(s,'visible','on');
      end
     case 'isodelete'
%删除 iso 对象
      [ a s] =getarrowslice;
      if numel(a)==1
        delete(getappdata(a,'arrowedge'));
      end
      cap=getappdata(s,'sliceomaticisocap');
      if ~isempty(cap)
        delete(cap);
      end
      delete(s);
      delete(a);
%设置 isosurface 对象的光照属性
     case 'isoflatlight'
      [ a s] =getarrowslice;
      set(s,'facelighting','flat');
     case 'isosmoothlight'
      [ a s] =getarrowslice;
      set(s,'facelighting','phong');
%设置 isosurface 对象的颜色属性
     case 'isocolor'
      [ a s] =getarrowslice;
      c=uisetcolor(get(s,'facecolor'));
      slowset(s,'facecolor',c,d.animincrement);
%设置 isosurface 对象的透明属性
     case 'isoalpha'
      [ a s] =getarrowslice;
      if nargin ~= 2
        error('Not enough arguments to sliceomatic.');
      end
      slowset(s,'facealpha',eval(p2),d.animincrement);
%设置 isocap 对象的属性
     case 'isocaps'
      [ a s] =getarrowslice;
      cap=getappdata(s,'isosurfacecap');
      if isempty(cap)
        new=localisocaps(s);
        set(new,'uicontextmenu',d.uiciso);
      else
        delete(cap);
        setappdata(s,'isosurfacecap',[]);
      end
```

```
        % slice现场菜单子选项
      case 'togglevisible'
       [a s]=getarrowslice;
       switch get(s,'visible')
        case 'on'
         set(s,'visible','off');
         pushset(a,'facealpha',.2);
        case 'off'
         set(s,'visible','on');
         popset(a,'facealpha');
       end
%设置slice对象的颜色属性
      case 'setfaceted'
       [a s]=getarrowslice;
       set(s,'edgec','k','facec','flat');
      if ischar(get(s,'facea')) && strcmp(get(s,'facea'),'texturemap')
         set(s,'facea','flat');
       end
       textureizeslice(s,'off');
      case 'setflat'
       [a s]=getarrowslice;
       set(s,'edgec','n','facec','flat');
      if ischar(get(s,'facea')) && strcmp(get(s,'facea'),'texturemap')
         set(s,'facea','flat');
       end
       textureizeslice(s,'off');
      case 'setinterp'
       [a s]=getarrowslice;
       set(s,'edgec','n','facec','interp');
      if ischar(get(s,'facea')) && strcmp(get(s,'facea'),'texturemap')
         set(s,'facea','interp');
       end
       textureizeslice(s,'off');
%设置slice对象的材质属性
      case 'settexture'
       [a s]=getarrowslice;
       set(s,'facecolor','texture','edgec','none');
       if ischar(get(s,'facea'))
         set(s,'facealpha','texturemap');
       end
       textureizeslice(s,'on');
      case 'setnone'
       [a s]=getarrowslice;
       set(s,'facecolor','none','edgec','none');
       textureizeslice(s,'off');
%设置slice对象的透明属性
      case 'setalphanone'
       [a s]=getarrowslice;
       slowset(s,'facealpha',1,d.animincrement);
      case 'setalphapoint5'
       [a s]=getarrowslice;
       slowset(s,'facealpha',.5,d.animincrement);
      case 'setalphaflat'
       [a s]=getarrowslice;
       set(s,'facealpha','flat');
      if ischar(get(s,'facec')) && strcmp(get(s,'facec'),'texturemap')
```

```
      set(s,'facecolor','flat');
      textureizeslice(s,'off');
    end
  case 'setalphainterp'
    [a s]=getarrowslice;
    set(s,'facealpha','interp');
   if ischar(get(s,'facec')) && strcmp(get(s,'facec'),'texturemap')
      set(s,'facecolor','interp');
      textureizeslice(s,'off');
    end
  case 'setalphatexture'
    [a s]=getarrowslice;
    set(s,'facealpha','texturemap');
    if ischar(get(s,'facec'))
      set(s,'facecolor','texturemap');
      textureizeslice(s,'on');
    end
%设置slice对象的等高线属性
  case 'slicecontour'
    [a s]=getarrowslice;
    localcontour(s, getappdata(s,'contour'));
  case 'slicecontourfullauto'
    [a s]=getarrowslice;
    d = getappdata(gcf, 'sliceomatic');
    minmax = get(d.axiso,'clim');
    levels = minmax(1):(minmax(2)-minmax(1))/10:minmax(2);
    setappdata(s, 'contourlevels', levels);
    localcontour(s, getappdata(s,'contour'),levels);
  case 'slicecontour_setauto'
    [a s]=getarrowslice;
    setappdata(s, 'contourlevels', []);
    localcontour(s, getappdata(s,'contour'));
%设置slice对象的等高线属性，包含等高线的高度属性
  case 'slicecontour_setfullauto'
    [a s]=getarrowslice;
    minmax = get(d.axiso,'clim');
    levels = minmax(1):(minmax(2)-minmax(1))/10:minmax(2);
    setappdata(s, 'contourlevels', levels);
    localcontour(s, getappdata(s,'contour'),levels);
  case 'slicecontour_select'
    [a s]=getarrowslice;
    d = getappdata(gcf, 'sliceomatic');
    xl = get(d.axiso,'xlim');
    levels = selectcontourlevels(get(s,'cdata'), xl(1), xl(2));
    setappdata(s, 'contourlevels', levels);
    localcontour(s, getappdata(s,'contour'),levels);
  case 'slicecontour_setlevels'
    [a s]=getarrowslice;
    d = getappdata(gcf, 'sliceomatic');
    xl = get(d.axiso,'xlim');
    levels = selectcontourlevels(get(s,'cdata'), xl(1), xl(2));
    setappdata(s, 'contourlevels', levels);
    localcontour(s, getappdata(s,'contour'),levels);
%删除slice对象的子菜单选项
  case 'deleteslice'
    [a s]=getarrowslice;
```

```
        if prod(size(a))==1
          delete(getappdata(a,'arrowedge'));
        end
        if ~isempty(getappdata(s,'contour'))
          delete(getappdata(s,'contour'));
        end
        delete(s);
        delete(a);
%删除slice对象的等高线子菜单选项
       case 'deleteslicecontour'
        [a s]=getarrowslice;
        if ~isempty(getappdata(s,'contour'))
          delete(getappdata(s,'contour'));
        end
        temp=getappdata(s);
        try
          temp.contourlevels;
          setappdata(s,'contourlevels',[]);
        end
        setappdata(s,'contour',[]);
%设置slice对象的等高线颜色属性
       case 'slicecontourflat'
        [a s]=getarrowslice;
        c = getappdata(s,'contour');
        if ~isempty(c)
          set(c,'edgecolor','flat');
        end
       case 'slicecontourinterp'
        [a s]=getarrowslice;
        c = getappdata(s,'contour');
        if ~isempty(c)
          set(c,'edgecolor','interp');
        end
       case 'slicecontourblack'
        [a s]=getarrowslice;
        c = getappdata(s,'contour');
        if ~isempty(c)
          set(c,'edgecolor','black');
        end
       case 'slicecontourwhite'
        [a s]=getarrowslice;
        c = getappdata(s,'contour');
        if ~isempty(c)
          set(c,'edgecolor','white');
        end
%设置slice对象的等高线直线属性
       case 'slicecontoursmooth'
        [a s]=getarrowslice;
        c = getappdata(s,'contour');
        onoff = get(gcbo,'checked');
        switch onoff
         case 'off'
          set(c,'linesmoothing','on');
         case 'on'
          set(c,'linesmoothing','off');
        end
```

```
case 'slicecontourcolor'
 [a s]=getarrowslice;
 c = getappdata(s,'contour');
 if ~isempty(c)
   inputcolor = get(c,'edgecolor');
   if isa(inputcolor,'char')
     inputcolor=[ 1 1 1 ];
   end
  slowset(c,'edgecolor',uisetcolor(inputcolor),d.animincrement);
 end
case 'slicecontourlinewidth'
 [a s]=getarrowslice;
 c = getappdata(s,'contour');
 if ~isempty(c)
   if isa(p2,'char')
     slowset(c,'linewidth',str2num(p2),d.animincrement);
   else
     slowset(c,'linewidth',p2,d.animincrement);
   end
 end
 %
 % All Slices 菜单的回调函数
 %
case 'allfacet'
 s=allSlices;
 set(s,'facec','flat','edgec','k');
 textureizeslice(s,'off');
case 'allflat'
 s=allSlices;
 set(s,'facec','flat','edgec','none');
 textureizeslice(s,'off');
case 'allinterp'
 s=allSlices;
 set(s,'facec','interp','edgec','none');
 textureizeslice(s,'off');
case 'alltex'
 s=allSlices;
 set(s,'facec','texturemap','edgec','none');
 textureizeslice(s,'on');
case 'allnone'
 s=allSlices;
 set(s,'facec','none','edgec','none');
 textureizeslice(s,'off');
case 'alltnone'
 s=allSlices;
 set(s,'facea',1);
 textureizeslice(s,'off');
case 'alltp5'
 s=allSlices;
 set(s,'facea',.5);
 textureizeslice(s,'off');
case 'alltflat'
 s=allSlices;
 set(s,'facea','flat');
 textureizeslice(s,'off');
case 'alltinterp'
```

```
 s=allSlices;
 set(s,'facea','interp');
 textureizeslice(s,'off');
case 'allttex'
 s=allSlices;
 set(s,'facea','texturemap');
 textureizeslice(s,'on');
 % 设置菜单的默认属性
 % 设置照明菜单的默认属性
case   'defaultfaceted'
 d.defcolor='faceted';
case   'defaultflat'
 d.defcolor='flat';
case   'defaultinterp'
 d.defcolor='interp';
case   'defaulttexture'
 d.defcolor='texture';
 if strcmp(d.defalpha,'flat') || strcmp(d.defalpha,'interp')
   d.defalpha='texture';
 end
case   'defaultinterp'
 d.defcolor='none';
 % 设置透明菜单的默认属性
case   'defaulttransnone'
 d.defalpha='none';
case   'defaulttransflat'
 d.defalpha='flat';
case   'defaulttransinterp'
 d.defalpha='interp';
case   'defaulttranstexture'
 d.defalpha='texture';
 d.defcolor='texture';
 % 设置光照菜单的默认属性
case      'defaultlightflat'
 d.deflight='flat';
case  'defaultlightsmooth'
 d.deflight='smooth';
 % 设置等高线菜单的默认属性
case 'defaultcontoursmooth'
 d.defaultcontoursmooth='on';
case 'defaultcontourflat'
 d.defcontourcolor='flat';
case 'defaultcontourinterp'
 d.defcontourcolor='interp';
case 'defaultcontourblack'
 d.defcontourcolor='black';
case 'defaultcontourwhite'
 d.defcontourcolor='white';
case 'defaultcontourlinewidth'
 if isa(p2,'char')
   d.defcontourlinewidth=str2num(p2);
 else
   d.defcontourlinewidth=p2;
 end
 % 显示camera工具栏
case 'cameratoolbar'
```

```
   cameratoolbar('Toggle');
  case 'annotationtoolbar'
   if propcheck(d.toolbar,'visible','on')
     set(d.toolbar,'vis','off');
   else
     set(d.toolbar,'vis','on');
   end
   % 控件属性
  case 'controlalpha'
   val=str2num(p2);
   iso=findobj(d.axiso,'type','image');
   if val == 0
     set([ d.pxx d.pxy d.pxz iso],'visible','off');
   else
     set([ d.pxx d.pxy d.pxz iso],'visible','on');
    slowset([ d.pxx d.pxy d.pxz] , 'facealpha',val,d.animincrement);
     slowset(iso,'alphadata',val,d.animincrement);
   end
  case 'toggleanimation'
   if d.animincrement == 0
     d.animincrement = 10;
   else
     d.animincrement = 0;
   end
  case 'controllabels'
   l = get(d.axx,'xticklabel');
   if isempty(l)
     set([ d.axx d.axiso],'xticklabelmode','auto');
     set([ d.axy d.axz],'yticklabelmode','auto');
   else
     set([ d.axx d.axiso],'xticklabel',[]);
     set([ d.axy d.axz],'yticklabel',[]);
   end
  case 'controlvisible'
   objs=findobj([ d.axiso d.axx d.axy d.axz]);
   if strcmp(get(d.axx,'visible'),'on')
     set(objs,'visible','off');
     set(d.axmain,'pos',[ .1 .1 .9 .8]);
   else
     set(objs,'visible','on');
     set(d.axmain,'pos',[ .2  .2 .6 .6]);
   end
   %
   % UICONTROL 回调函数
   %
  case 'colormap'
   str=get(gcbo,'string');
   val=str{ get(gcbo,'value')};
   size(val);
   if strcmp(val,'custom')
     cmapeditor
   else
     slowset(gcf,'colormap',feval(val),d.animincrement);
   end
  case 'alphamap'
   str=get(gcbo,'string');
```

```
        val=alphamap(str( get(gcbo,'value')));
        slowset(gcf,'alphamap',val,d.animincrement);
         % Commands
       case 'copy'
        copyobj(gca,figure);set(gca,'pos',[.1 .1 .9 .8]);
       case 'print'
        newf=figure('visible','off','renderer',get(gcf,'renderer'));
        copyobj(d.axmain,newf);
        set(gca,'pos',[ .1 .1 .9 .8])
        printdlg(newf);
        close(newf);
       otherwise
        error('Bad slice-o-matic command.');
      end
    catch
      disp(get(0,'errormessage'));
    end
    setappdata(gcf,'sliceomatic',d);
  else
    disp('Sliceomatic data must be DOUBLE');
  end
```

上面程序是该GUI对象的主程序代码，因此显得比较冗长，但是并不复杂，结构也十分清晰。首先调用sliceomaticsetdata函数为图形对象准备数据，然后调用sliceomaticfigure命令绘制主要的GUI界面和主要控件，根据sliceomaticfigure中各个参数的不同取值，调用对应的函数命令，因此，可以看到上面的程序代码中存在着大量的case…end和if…elseif…end循环分支结构。



限于篇幅，关于各循环结构中程序代码的具体含义，在本小节中就不详细介绍了，请读者自行分析。当读者完成上面的所有程序后，将代码保存为"sliceomatic.m"文件，该文件将是GUI对象的主要程序代码。

11.9.8 设置GUI对象的菜单选项

延续上面小节的步骤。

step 1 在上面步骤打开的M文件编辑器中，输入下面的代码：

```
%------------------------------------------------------------------------
function fileName = UserStickyPrefsFileName
localPath = fileparts(which(mfilename));
fileName = fullfile(localPath,'Sliceomatic.Prefs.mat');
```

上面的程序代码的功能是，通过fileparts命令获取M文件的路径名称，然后使用fullfile命令返回完整的文件名称。



在MATLAB中，fileparts命令的功能是返回M文件的部分名称，其完整的调用方式为[pathstr,name,ext,versn] = fileparts('filename')，其中pathstr表示文件保存路径，name表示文件名称，ext表示文件后缀。

step 2 在上面步骤打开的M文件编辑器中，输入下面的代码：

```
%-----------------------------------------------------------------------
function dOut = OverrideStickyUserPreferences(d)
%characteristic prefs file (stored locally where Slice-O-Matic installed)
fileName = UserStickyPrefsFileName;
% 加载属性数值
if exist(fileName,'file')
  load(fileName)
  set(d.toolbar,'visible',prefs.anntoolbar_Checked)
  if prefs.camtoolbar_checked
    cameratoolbar('show');
  else
    cameratoolbar('hide');
  end
% 读取对应的属性数值
  d.defcolor = prefs.defcolor;
  d.defalpha = prefs.defalpha;
  d.deflight = prefs.deflight;
  d.defcontourcolor = prefs.defcontourcolor;
  d.defcontourlinewidth = prefs.defcontourlinewidth;
  d.defcontoursmooth = prefs.defcontoursmooth;
% 判断图形的坐标轴标签属性模式
    if strcmp('auto',prefs.ticklabels)
    set([ d.axx d.axiso] ,'xticklabelmode','auto');
    set([ d.axy d.axz] ,'yticklabelmode','auto');
  else
    set([ d.axx d.axiso] ,'xticklabel',[] );
    set([ d.axy d.axz] ,'yticklabel',[] );
  end
  d.animincrement = prefs.animincrement;
  set([ d.pxx d.pxy d.pxz]  , 'facealpha',prefs.controlalpha);
  iso = findobj(d.axiso,'type','image');
  set(iso,'alphadata',prefs.controlalpha);
% 显示已经加载属性列表数值
  disp('Sticky preferences loaded.')
end
% 返回编辑后的图形属性结构体变量
dOut = d;
```

上面的程序代码功能是覆盖用户自行设置的对象属性，在程序代码的开始，首先加载文件中的数据，然后设置图形界面中的对象属性。

step 3 在上面步骤打开的 M 文件编辑器中，输入下面的代码：

```
%-----------------------------------------------------------------------
function SavePrefs(obj,event)
% 保存关于对象属性的结构体
d = getappdata(gcf,'sliceomatic');
%extract only preferences that need to be sticky
prefs.anntoolbar_Checked = get(d.toolbar,'Visible');
prefs.defcolor = d.defcolor;
prefs.defalpha = d.defalpha;
prefs.deflight = d.deflight;
prefs.defcontourcolor = d.defcontourcolor;
prefs.defcontourlinewidth = d.defcontourlinewidth;
prefs.defcontoursmooth = d.defcontoursmooth;
prefs.camtoolbar_checked = cameratoolbar('getvisible');
```

```
prefs.ticklabels = get(d.axx,'xticklabelmode');
prefs.animincrement = d.animincrement;
prefs.controlalpha = get(d.pxx,'facealpha');
% 将属性列表数值保存到对应的文件中
fileName = UserStickyPrefsFileName;
save(fileName,'prefs')
disp([ 'Saved: ' fileName])
```

上面的程序代码设置的是该GUI对象保存文件的属性，当在程序代码中调用该程序时，将会直接设置所有的保存信息。

step 4 在上面步骤中打开的 M 文件编辑器中，输入下面的代码：

```
function controlmenu(fig, action)
% 处理关于控制菜单的程序代码
   d=getappdata(gcf,'sliceomatic');
  if cameratoolbar('getvisible')
    set(d.camtoolbar,'checked','on');
  else
    set(d.camtoolbar,'checked','off');
  end
    if exist('uitoolfactory') == 2
    if propcheck(d.toolbar,'visible','on')
      set(d.anntoolbar,'checked','on');
    else
      set(d.anntoolbar,'checked','off');
    end
  end
     set([ d.dcalpha1 d.dcalpha8 d.dcalpha6 d.dcalpha5 d.dcalpha6 d.
dcalpha2 d.dcalpha0...
        d.dclabels d.dcvis ],...
       'checked','off');
  switch get(d.pxx,'facealpha')
   case 1, set(d.dcalpha1,'checked','on');
   case .8, set(d.dcalpha8,'checked','on');
   case .6, set(d.dcalpha6,'checked','on');
   case .5, set(d.dcalpha5,'checked','on');
   case .4, set(d.dcalpha4,'checked','on');
   case .2, set(d.dcalpha2,'checked','on');
   case 0, set(d.dcalpha0,'checked','on');
  end
  if d.animincrement == 0
    set(d.dcanimstep,'checked','off');
  else
    set(d.dcanimstep,'checked','on');
  end
    if ~isempty(get(d.axx,'xticklabel'))
    set(d.dclabels,'checked','on');
  end
    if strcmp(get(d.axx,'visible'),'on')
    set(d.dcvis,'checked','on');
  end
    if 0
    xt = get(get(d.axx,'title'),'string');
    switch xt
     case 'X Slice Controller'
```

```
      set(d.dcslice,'checked','on');
    end
    xt = get(get(d.axiso,'title'),'string');
    switch xt
     case 'Iso Surface Controller'
      set(d.dciso,'checked','on');
    end
  end
```

上面程序代码的主要功能是设置“Controls”菜单选项的各种属性，首先设置照明工具栏和注释工具栏中的按钮的检录属性“checked”的数值。在MATLAB中，菜单选项的检录属性的取值有on和off两个。默认情况下，菜单选项的“checked”属性数值为“off”，不会显示检录符；当其属性的数值为“on”时，用户选中该菜单选项，其选项就会出现“√”标记。在本实例中，“Controls”菜单选项的主要功能是控制工具栏的显示选项和图形对象的控制选项（透明度设置）等。

step 5 在上面步骤打开的M文件编辑器中，输入下面的代码：

```
function defaultmenu(fig, action)
% Handle toggling bits on the slice defaults menu
    d=getappdata(gcf,'sliceomatic');
    set([ d.dfacet d.dflat d.dinterp d.dtex d.dtnone d.dtflat d.dtinterp ...
          d.dttex d.dcflat d.dcinterp d.dcblack d.dcwhite d.dcnone ...
          d.dlflat d.dlsmooth ...
         d.smcl1 d.smcl2 d.smcl3 d.smcl4 d.smcl5 d.smcl6 ], 'checked','off');
  switch d.defcolor
   case 'faceted'
    set(d.dfacet,'checked','on');
   case 'flat'
    set(d.dflat,'checked','on');
   case 'interp'
    set(d.dinterp,'checked','on');
   case 'texture'
    set(d.dtex,'checked','on');
   case 'none'
    set(d.dcnone,'checked','on');
  end
  switch d.defalpha
   case 'none'
    set(d.dtnone,'checked','on');
   case 'flat'
    set(d.dtflat,'checked','on');
   case 'interp'
    set(d.dtinterp,'checked','on');
   case 'texture'
    set(d.dttex,'checked','on');
  end
  switch d.deflight
   case 'flat'
    set(d.dlflat,'checked','on');
   case 'smooth'
    set(d.dlsmooth,'checked','on');
  end
  switch d.defcontourcolor
```

```
    case 'flat'
     set(d.dcflat,'checked','on');
    case 'interp'
     set(d.dcinterp,'checked','on');
    case 'black'
     set(d.dcblack,'checked','on');
    case 'white'
     set(d.dcwhite,'checked','on');
   end
   %set(d.dcsmooth,'checked',d.defcontoursmooth);
   switch d.defcontourlinewidth
    case 1, set(d.dcl1,'checked','on');
    case 2, set(d.dcl2,'checked','on');
    case 3, set(d.dcl3,'checked','on');
    case 4, set(d.dcl4,'checked','on');
    case 5, set(d.dcl5,'checked','on');
    case 6, set(d.dcl6,'checked','on');
   end
```

上面程序代码的主要功能是设置“Object_Defaults”菜单选项的各种属性。在本GUI实例中，“Object_Defaults”菜单选项提供用户设置Slice截面的颜色、透明度，Isosurface曲面的光照和等高线的颜色等属性。用户可以选择对应的菜单选项，设置这些图形对象的默认属性。当用户选中对应选项后，其选项就会出现“√”标记。

step 6 在上面步骤打开的M文件编辑器中，输入下面的代码：

```
function slicecontextmenu(fig,action)
% Context menu state for slices
  d=getappdata(gcf,'slioeomatic');
  [a s]=getarrowslice;
  set([ d.smfacet d.smflat d.sminterp d.smtex d.smtnone d.smtp5 ...
       d.smtflat d.smtinterp d.smttex d.smnone d.smcsmooth
     ],'checked','off');
  set(d.vistog,'checked',get(s,'visible'));
    if propcheck(s,'edgec',[0 0 0])
    set(d.smfacet,'checked','on');
  elseif propcheck(s,'facec','flat')
    set(d.smflat,'checked','on');
  end
  if propcheck(s,'facec','interp')
    set(d.sminterp,'checked','on');
  end
  if propcheck(s,'facec','texturemap')
    set(d.smtex,'checked','on');
  end
  if propcheck(s,'facec','none')
    set(d.smnone,'checked','on');
  end
  if propcheck(s,'facea',1)
    set(d.smtnone,'checked','on');
  end
  if propcheck(s,'facea',.5)
    set(d.smtp5,'checked','on');
  end
  if propoheck(s,'facea','flat')
```

```
  set(d.smtflat,'checked','on');
end
if propcheck(s,'facea','interp')
  set(d.smtinterp,'checked','on');
end
if propcheck(s,'facea','texturemap')
  set(d.smttex,'checked','on');
end
cm = [ d.smcflat d.smcinterp d.smcblack d.smcwhite d.smccolor ...
     d.smcl1 d.smcl2 d.smcl3 d.smcl4 d.smcl5 d.smcl6 ];
set(cm,'checked','off');
if isempty(getappdata(s,'contour'))
  set(d.smcontour,'enable','on');
  set(d.smcsetauto,'enable','off');
  set(d.smcsetav,'enable','off');
  set(d.smclevels,'enable','off');
  set(d.smrcontour,'enable','off');
  set(d.smcsmooth,'enable','off');
  set(cm,'enable','off');
else
  set(d.smcontour,'enable','off')
  set(d.smcsetauto,'enable','on');
  set(d.smcsetav,'enable','on');
  set(d.smclevels,'enable','on');
  set(d.smrcontour,'enable','on')
  set(d.smcsmooth,'enable','on');
  set(cm,'enable','on')
  c = getappdata(s,'contour');
  if propcheck(c,'linesmoothing','on')
    set(d.smcsmooth,'checked','on');
  end
  ec = get(c,'edgecolor');
  if isa(ec,'char')
    switch ec
     case 'flat'
      set(d.smcflat,'checked','on');
     case 'interp'
      set(d.smcinterp,'checked','on');
    end
  else
    if ec == [ 1 1 1 ]
      set(d.smcwhite,'checked','on');
    elseif ec == [ 0 0 0 ]
      set(d.smcblack,'checked','on');
    else
      set(d.smccolor,'checked','on');
    end
  end
  clw = get(c,'linewidth');
  switch clw
   case 1, set(d.smcl1,'checked','on');
   case 2, set(d.smcl2,'checked','on');
   case 3, set(d.smcl3,'checked','on');
   case 4, set(d.smcl4,'checked','on');
   case 5, set(d.smcl5,'checked','on');
   case 6, set(d.smcl6,'checked','on');
```

```
    end
  end
```

上面程序代码的主要功能是设置关于Slice属性的现场菜单选项，用户可以设置Slice截面的颜色、透明度、照明处理、线宽等属性。上面程序代码的结构并不复杂，请用户自行分析其代码含义。

step 7 在上面步骤打开的M文件编辑器中，输入下面的代码：

```
function isocontextmenu(fig,action)
% Context menu state for isosurfaces
  d=getappdata(gcf,'sliceomatic');
    [a s]=getarrowslice;
  if propcheck(s,'facelighting','flat')
    set(d.isoflatlight,'checked','on');
    set(d.isosmoothlight,'checked','off');
  else
    set(d.isoflatlight,'checked','off');
    set(d.isosmoothlight,'checked','on');
  end
  set(d.vistogiso,'checked',get(s,'visible'));
  if ~isempty(getappdata(s,'isosurfacecap'))
    set(d.isocap,'checked','on');
  else
    set(d.isocap,'checked','off');
  end
```

上面程序代码的主要功能是设置关于Isosurface曲面属性的现场菜单选项，用户可以设置该曲面的光照属性等。

step 8 在上面步骤打开的M文件编辑器中，输入下面的代码：

```
function outd = figmenus(d)
% Set up sliceomatic's gui menus within structure D
% Main Figure Menu
  set(gcf,'menubar','none');

% File menu
  d.filemenu = uimenu(gcf,'label','File');
   d.fcopy = uimenu(d.filemenu, 'label', 'Copy figure','callback',
'sliceomatic copy');
   d.fprint  = uimenu(d.filemenu,'label','Print...','callback',
'sliceomatic print');
  %%% start patch 1of3 RAB 2/18/05 %%%
  d.fsaveprefs = uimenu(d.filemenu,'label','Save preferences','callback',
@SavePrefs);
  %%% end patch 1of3 RAB 2/18/05 %%%
  % How do get these props onto the print figure?
  %d.fprints = uimenu(d.filemenu,'label','Print Setup...','callback',
'printdlg -setup');
  % ---
  d.fexit = uimenu(d.filemenu, 'label', 'Close','callback','closereq',...
                   'separator','on');
% Controls Menu
   d.defcontrols = uimenu(gcf,'label','Controls', 'callback',
```

```
@controlmenu);
  if exist('uitoolfactory') == 2
    d.anntoolbar = uimenu(d.defcontrols,'label','Annotations toolbar',
'callback', 'sliceomatic annotationtoolbar');
  end
  d.camtoolbar = uimenu(d.defcontrols,'label','Camera toolbar',
'callback', 'sliceomatic cameratoolbar');
  d.dcalpha = uimenu(d.defcontrols,'label','Controls Transparency');
  d.dcalpha1= uimenu(d.dcalpha,'label','1','callback','sliceomatic
controlalpha 1');
  d.dcalpha8= uimenu(d.dcalpha,'label','.8','callback','sliceomatic
controlalpha .8');
  d.dcalpha6= uimenu(d.dcalpha,'label','.6','callback','sliceomatic
controlalpha .6');
  d.dcalpha5= uimenu(d.dcalpha,'label','.5','callback','sliceomatic
controlalpha .5');
  d.dcalpha4= uimenu(d.dcalpha,'label','.4','callback','sliceomatic
controlalpha .4');
  d.dcalpha2= uimenu(d.dcalpha,'label','.2','callback','sliceomatic
controlalpha .2');
  d.dcalpha0= uimenu(d.dcalpha,'label','0','callback','sliceomatic
controlalpha 0');
  d.dcanimstep = uimenu(d.defcontrols,'label','Animation','callback',
'sliceomatic toggleanimation');
  d.dclabels= uimenu(d.defcontrols','label','Tick Labels','callback',
'sliceomatic controllabels');
  d.dcvis    = uimenu(d.defcontrols','label','Visible','callback',
'sliceomatic controlvisible');
  % d.dsetrange= uimenu(d.defcontrols','label','Set Range','callback',
'@setvolumerange');
  % d.dcslice = uimenu(d.defcontrols,'label','Slice Controls','callback',
'sliceomatic useslicecontrols');
  % d.dciso    = uimenu(d.defcontrols,'label','Iso Surface Control',
'callback','sliceomatic useisocontrols','separator','on');

  % Remove this once we have more controls to enable and disable.
  %  set(d.defcontrols,'vis','off');

  % Default for new slices menu
  d.defmenu = uimenu(gcf,'label','Object_Defaults', 'callback',
@defaultmenu);
  d.dfacet  = uimenu(d.defmenu,'label','Slice Color Faceted','callback',
'sliceomatic defaultfaceted');
  d.dflat   = uimenu(d.defmenu,'label','Slice Color Flat',   'callback',
'sliceomatic defaultflat');
  d.dinterp = uimenu(d.defmenu,'label','Slice Color Interp', 'callback',
'sliceomatic defaultinterp');
  d.dtex    = uimenu(d.defmenu,'label','Slice Color Texture','callback',
'sliceomatic defaulttexture');
  d.dcnone  = uimenu(d.defmenu,'label','Slice Color None','callback',
'sliceomatic defaultcolornone');
  d.dtnone   = uimenu(d.defmenu,'label','Slice Transparency None',
'callback','sliceomatic defaulttransnone','separator','on');
  d.dtflat   = uimenu(d.defmenu,'label','Slice Transparency Flat',
'callback','sliceomatic defaulttransflat');
  d.dtinterp= uimenu(d.defmenu,'label','Slice Transparency Interp',
```

```
'callback','sliceomatic defaulttransinterp');
  d.dttex    = uimenu(d.defmenu,'label','Slice Transparency Texture',
'callback','sliceomatic defaulttranstexture');
  d.dlflat   = uimenu(d.defmenu,'label','IsoSurface Lighting Flat',
'callback','sliceomatic defaultlightflat','separator','on');
  d.dlsmooth= uimenu(d.defmenu,'label','IsoSurface Lighting Smooth',
'callback','sliceomatic defaultlightsmooth');
  %d.dcsmooth= uimenu(d.defmenu,'label','Contour Line Smoothing',
'callback','sliceomatic defaultcontoursmooth');
  d.dcflat  = uimenu(d.defmenu,'label','Contour Color Flat',   'callback',
'sliceomatic defaultcontourflat','separator','on');
  d.dcinterp= uimenu(d.defmenu,'label','Contour Color Interp',
'callback','sliceomatic defaultcontourinterp');
  d.dcblack = uimenu(d.defmenu,'label','Contour Color Black',  'callback',
'sliceomatic defaultcontourblack');
  d.dcwhite = uimenu(d.defmenu,'label','Contour Color White',  'callback',
'sliceomatic defaultcontourwhite');
  d.dclinew = uimenu(d.defmenu,'label','Contour Line Width');
  d.dcl1     = uimenu(d.dclinew,'label','1','callback','sliceomatic
defaultcontourlinewidth 1');
  d.dcl2     = uimenu(d.dclinew,'label','2','callback','sliceomatic
defaultcontourlinewidth 2');
  d.dcl3     = uimenu(d.dclinew,'label','3','callback','sliceomatic
defaultcontourlinewidth 3');
  d.dcl4     = uimenu(d.dclinew,'label','4','callback','sliceomatic
defaultcontourlinewidth 4');
  d.dcl5     = uimenu(d.dclinew,'label','5','callback','sliceomatic
defaultcontourlinewidth 5');
  d.dcl6     = uimenu(d.dclinew,'label','6','callback','sliceomatic
defaultcontourlinewidth 6');
    d.defcolor='texture';
  d.defalpha='texture';
  d.deflight='smooth';
  d.defcontourcolor='black';
  d.defcontourlinewidth=1;
  % This exposes an unpleasant R14 bug
  d.defcontoursmooth='off';
  % investigate hardware opengl.
  inc = 0;
  try
    od = opengl('data');
    if isfield(od,'Software')
      % R14 version of MATLAB
      if ~od.Software
        inc = 10;
      end
    else
      % Older version of MATLAB
      if ~(strcmp(od.Renderer,'Mesa X11') || ...
           strcmp(od.Renderer, 'GDI Generic'))
        inc = 10;
      end
    end
  end
  d.animincrement=inc;
    %%% start patch 2of3 RAB 2/18/05 %%%
```

```
d = OverrideStickyUserPreferences(d);
%%% end patch 2of3 RAB 2/18/05 %%%
  % Set props for all slices menu
d.allmenu = uimenu(gcf,'label','AllSlices');
uimenu(d.allmenu,'label','Color Faceted','callback','sliceomatic allfacet');
uimenu(d.allmenu,'label','Color Flat','callback','sliceomatic allflat');
uimenu(d.allmenu,'label','Color Interp','callback','sliceomatic allinterp');
uimenu(d.allmenu,'label','Color Texture','callback','sliceomatic alltex');
uimenu(d.allmenu,'label','Color None','callback','sliceomatic allnone');
uimenu(d.allmenu,'label','Transparency None','callback','sliceomatic alltnone','separator','on');
uimenu(d.allmenu,'label','Transparency .5','callback','sliceomatic alltp5');
uimenu(d.allmenu,'label','Transparency Flat','callback','sliceomatic alltflat');
uimenu(d.allmenu,'label','Transparency Interp','callback','sliceomatic alltinterp');
uimenu(d.allmenu,'label','Transparency Texture','callback','sliceomatic allttex');
% Setup Help style options
d.helpmenu = uimenu(gcf,'label','Help');
uimenu(d.helpmenu,'label','Help','callback','doc sliceomatic/sliceomatic');
uimenu(d.helpmenu,'label','Check for Updates','callback','web http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=764&objectType=FILE');
uimenu(d.helpmenu,'label','About Author','callbaok','web http://www.mathworks.com/matlabcentral/fileexchange/loadAuthor.do?objectId=803709&objectType=author');
% Context Menus
% Slice Context Menu
d.uic=uicontextmenu('callback', @slicecontextmenu);
d.vistog = uimenu(d.uic,'label','Visible','callback','sliceomatic togglevisible');
d.uicdelete = uimenu(d.uic,'label','Delete','callback','sliceomatic deleteslice');
d.smcolorm  = uimenu(d.uic,'label','Color','separator','on');
d.smfacet   = uimenu(d.smcolorm,'label','Color Faceted','callback','sliceomatic setfaoeted');
d.smflat    = uimenu(d.smcolorm,'label','Color Flat','callback','sliceomatic setflat');
d.sminterp  = uimenu(d.smcolorm,'label','Color Interp','callback','sliceomatic setinterp');
d.smtex     = uimenu(d.smcolorm,'label','Color Texture','callback','sliceomatic settexture');
d.smnone    = uimenu(d.smcolorm,'label','Color None','callback','sliceomatic setnone');
d.smtransm  = uimenu(d.uic,'label','Transparency');
d.smtnone   = uimenu(d.smtransm,'label','Transparency None','callback','sliceomatic setalphanone');
d.smtp5     = uimenu(d.smtransm,'label','Transparency .5','callback',
```

```
'sliceomatic setalphapoint5');
 d.smtflat   = uimenu(d.smtransm,'label','Transparency Flat','callback',
'sliceomatic setalphaflat');
  d.smtinterp = uimenu(d.smtransm,'label','Transparency Interp',
'callback','sliceomatic setalphainterp');
  d.smttex     = uimenu(d.smtransm,'label','Transparency Texture',
'callback','sliceomatic setalphatexture');
  d.smcontour = uimenu(d.uic,'label','Add Contour','separator','on');
  d.smcont0    = uimenu(d.smcontour,'label','Auto (Slice)','callback',
'sliceomatic slicecontour');
  d.smcont0v  = uimenu(d.smcontour,'label','Auto (Volume)','callback',
'sliceomatic slicecontourfullauto');
  d.smcont1    = uimenu(d.smcontour,'label','Select Levels','callback',
'sliceomatic slicecontour_select','separator','on');
   d.smcsetauto= uimenu(d.uic,'label','Set Auto Levels (Slice)',
'callback','sliceomatic slicecontour_setauto');
   d.smcsetav   = uimenu(d.uic,'label','Set Auto Levels (Volume)',
'callback','sliceomatic slicecontour_setfullauto');
  d.smclevels = uimenu(d.uic,'label','Set Levels','callback','sliceomatic
slicecontour_setlevels');
   d.smrcontour= uimenu(d.uic,'label','Remove Contour','callback',
'sliceomatic deleteslicecontour');
  d.smccm       = uimenu(d.uic,'label','Contour Colors');
   d.smcflat    = uimenu(d.smccm,'label','Contour Flat','callback',
'sliceomatic slicecontourflat');
   d.smcinterp = uimenu(d.smccm,'label','Contour Interp','callback',
'sliceomatic slicecontourinterp');
   d.smcblack   = uimenu(d.smccm,'label','Contour Black','callback',
'sliceomatic slicecontourblack');
   d.smcwhite   = uimenu(d.smccm,'label','Contour White','callback',
'sliceomatic slicecontourwhite');
   d.smccolor   = uimenu(d.smccm,'label','Contour Color','callback',
'sliceomatic slicecontourcolor');
   d.smcsmooth = uimenu(d.uic,'visible','off','label','Smooth Contour
Lines','callback','sliceomatic slicecontoursmooth');
   d.smclinew   = uimenu(d.uic,'label','Contour Line Width');
   d.smcl1       = uimenu(d.smclinew,'label','1','callback','sliceomatic
slicecontourlinewidth 1');
   d.smcl2       = uimenu(d.smclinew,'label','2','callback','sliceomatic
slicecontourlinewidth 2');
   d.smcl3       = uimenu(d.smclinew,'label','3','callback','sliceomatic
slicecontourlinewidth 3');
   d.smcl4       = uimenu(d.smclinew,'label','4','callback','sliceomatic
slicecontourlinewidth 4');
   d.smcl5       = uimenu(d.smclinew,'label','5','callback','sliceomatic
slicecontourlinewidth 5');
   d.smcl6       = uimenu(d.smclinew,'label','6','callback','sliceomatic
slicecontourlinewidth 6');
      % Isosurface Context Menu
   d.uiciso=uicontextmenu('callback',@isocontextmenu);
  d.vistogiso = uimenu(d.uiciso,'label','Visible','callback','sliceomatic
isotogglevisible');
  d.isodelete = uimenu(d.uiciso,'label','Delete','callback','sliceomatic
isodelete');
   d.isoflatlight=uimenu(d.uiciso,'label','LightingFlat','callback',
'sliceomatic oflatlight','separator','on');
```

```
  d.isosmoothlight=uimenu(d.uiciso,'label','Lighting Smooth','callback',
'sliceomatic isosmoothlight');
  d.isocolor = uimenu(d.uiciso,'label','Change Color','callback',
'sliceomatic isocolor','separator','on');
  d.isoalpha=uimenu(d.uiciso,'label','Change Transparency');
  uimenu(d.isoalpha,'label','.2','callback','sliceomatic isoalpha .2');
  uimenu(d.isoalpha,'label','.5','callback','sliceomatic isoalpha .5');
  uimenu(d.isoalpha,'label','.8','callback','sliceomatic isoalpha .8');
  uimenu(d.isoalpha,'label','1','callback','sliceomatic isoalpha 1');
  d.isocap=uimenu(d.uiciso,'label','Add IsoCaps','callback','sliceomatic
isocaps','separator','on');
  outd = d;
```

上面程序代码的主要功能是设置GUI图形界面的所有菜单选项，包括菜单栏菜单选项、各种现场菜单等，使用的就是uimenu命令，读者可以参考前面章节中的相关内容。最后，将本小节中的程序代码保存为“figmenus.m”文件。

11.9.9 检测程序代码

延续上面小节的步骤。

step 1 在MATLAB的命令窗口中输入下面的命令：

```
>> sliceomatic
```

step 2 查看图形结果。输入上面的命令，按'Enter'键后，显示下面的内容：

```
Smoothing for IsoNormals...
Generating reduction volume...
```

同时，MATLAB显示程序代码的默认GUI图形界面，如图11.104所示。

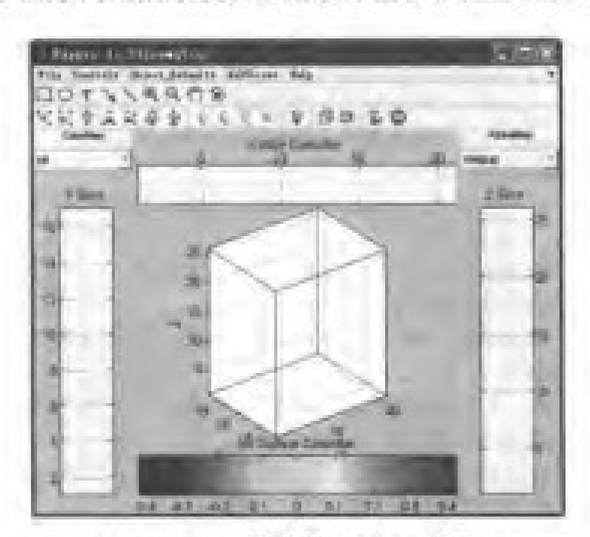


图11.104　默认的GUI图形界面

step 3 选择切片图的Z向坐标。在上面的GUI图形界面中，可以使用鼠标选择相应的切片面的Z向坐标数值，得到的结果如图11.105所示。

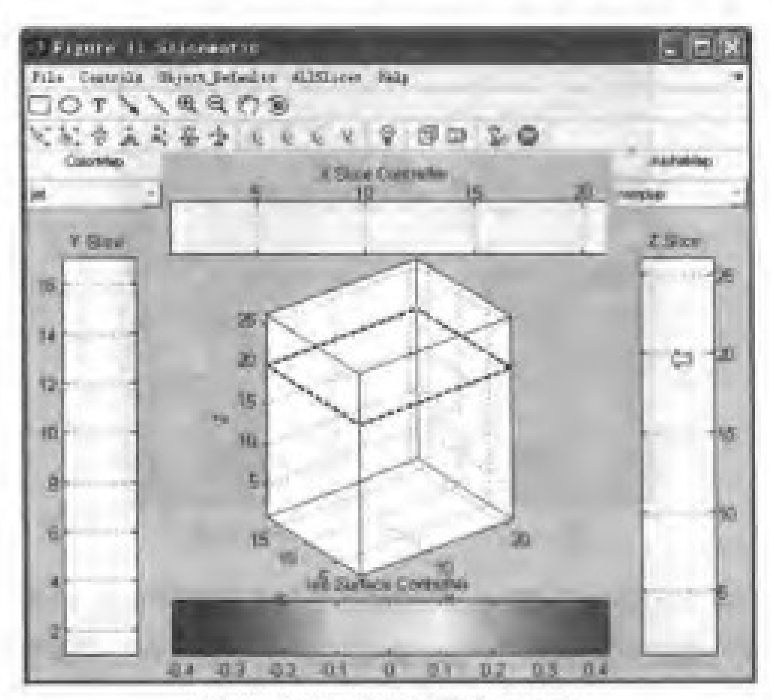


图 11.105　选择切面 Z 向坐标

从上面的图形结果中可以看出，当在“Z Slice”控件中选择 Slice 截面的 Z 坐标数值时，在“Z Slice”控件的对应坐标数值上会显示“箭头”光标对象，同时，在三维图形界面中显示以虚线表示的 Slice 截面。

step 4 查看切片面结果。当用户选定对应的 Z 向坐标后，单击鼠标右键，在“Z Slice”控件中会显示“箭头”光标对象，同时可以绘制截面图形，如图 11.106 所示。



当用户没有设置截面的相关属性（色图系统、透明度）时，程序代码会根据默认属性绘制该截面对象，可以使用工具栏或者菜单来修改这些属性。

step 5 选择其他坐标轴向的切面。重复上面的步骤，可以选择 Slice 截面的 X、Y 和 Z 向坐标，绘制各个方向上的 Slice 截面，得到的结果如图 11.107 所示。

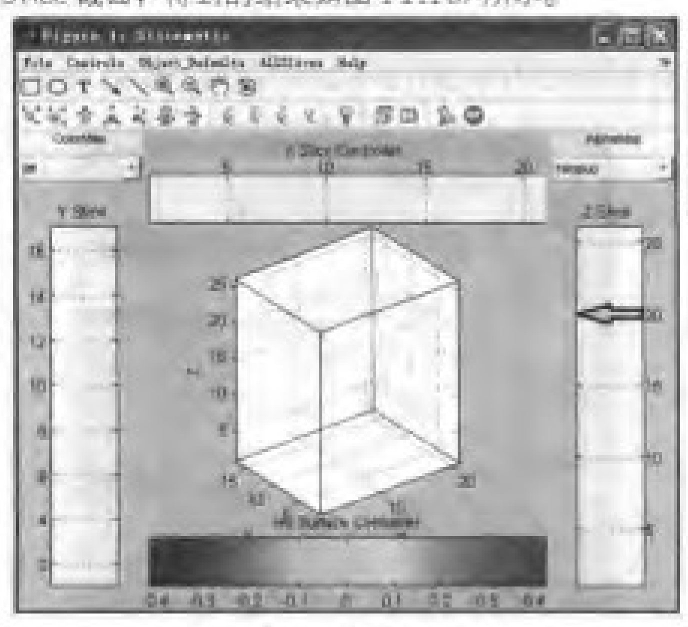


图 11.106　绘制 Slice 截面

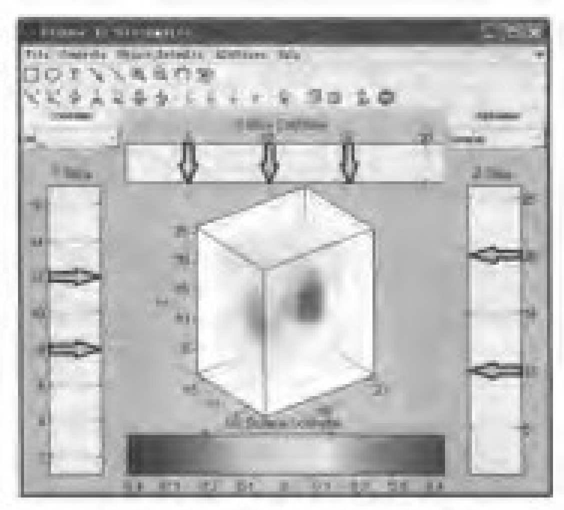


图 11.107 选择其他方向的截面

step 6 修改绘制图形的"colormap"属性。当绘制了各个方向的截面后，可以选择"colormap"下拉菜单中的选项，修改其色图颜色。在本步骤中选择的选项是"hot"，得到的结果如图11.108所示。

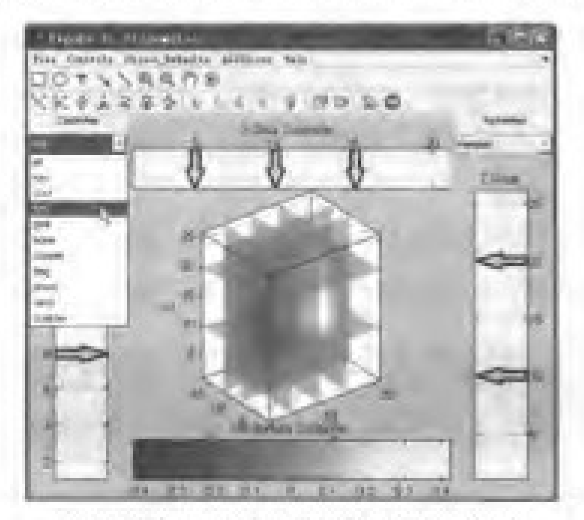


图 11.108 修改图形对象的色图颜色

step 7 修改绘制图形的"Alphamap"属性。可以选择"AlphaMap"下拉菜单中的选项修改图形对象的透明度属性。在本例中选择的透明度类型为"vdown"选项，得到的结果如图11.109所示。

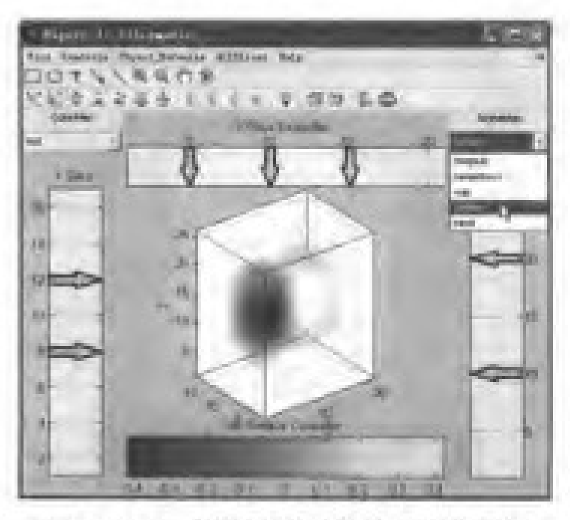


图 11.109 修改图形对象的透明度属性

step 8 修改图形对象的视角。单击“照明”工具栏中的相关按钮，修改图形对象的视角，得到的结果如图 11.110 所示。

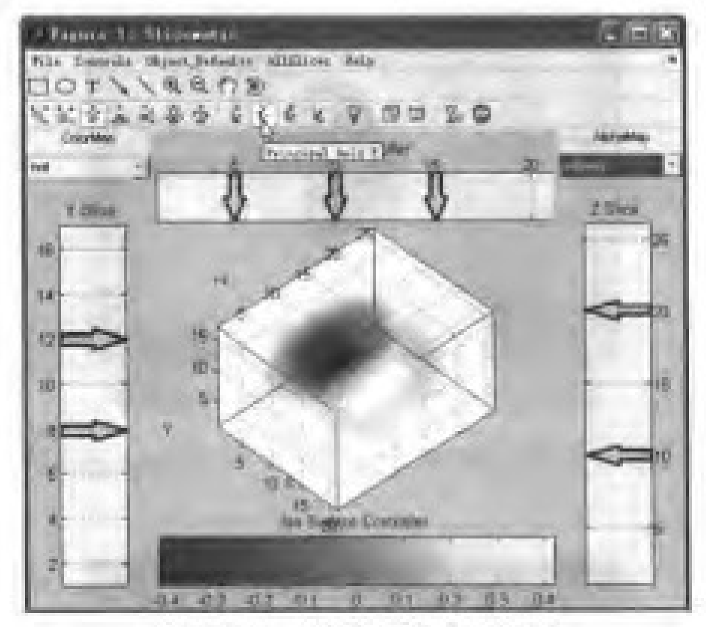


图 11.110 修改图形对象的视角

step 9 添加 Isosurface 曲面。选择 "Iso Surface Controller" 中的对应坐标数值，单击鼠标左键，在该坐标数值中添加 Isosurface 曲面，得到的结果如图 11.111 所示。



为了方便后续步骤的操作，在添加 Isosurface 曲面之前，通过使用“照明”工具栏中的相关按钮将图形的坐标轴转换为默认的三维坐标。

step 10 删除 Isosurface 曲面。选择上面添加的 Isosurface 曲面，然后单击鼠标右键，在弹出的现场菜单中选择 "Delete" 选项，删除添加的 Isosurface 曲面，如图 11.112 所示。

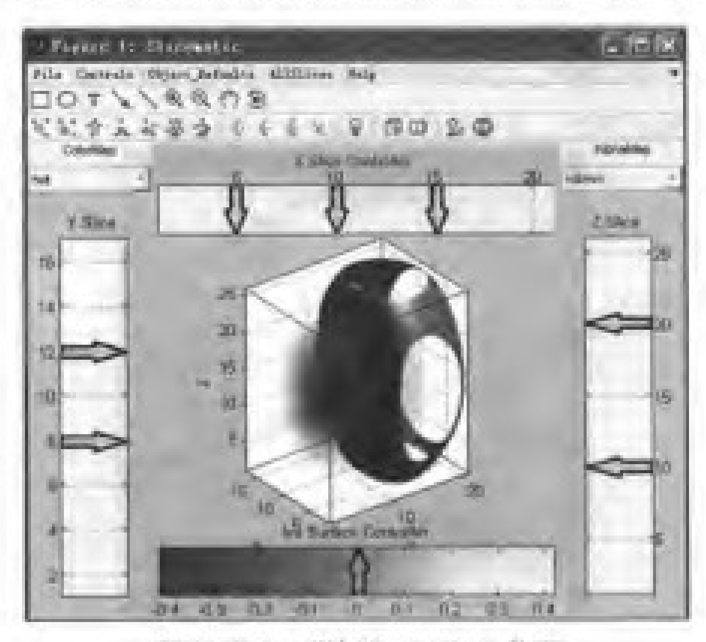
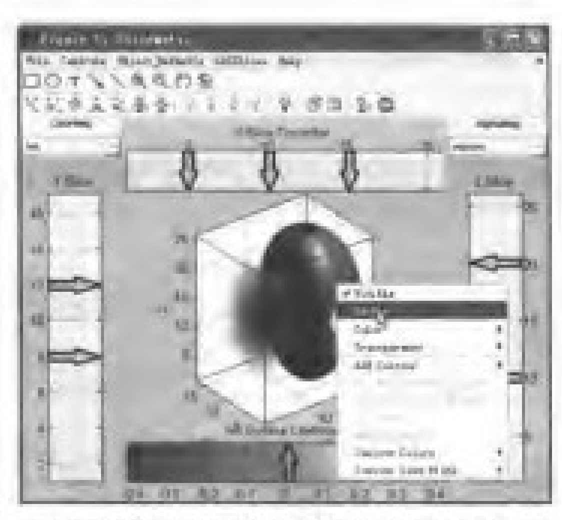
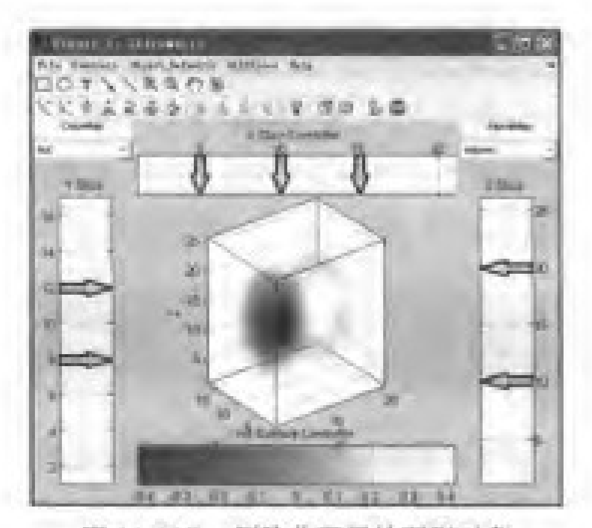


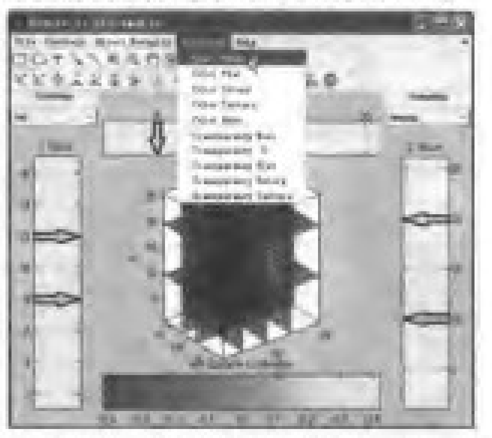
图 11.111 添加 Isosurface 曲面


图 11.112　删除添加的 Isosurface 曲面

step 11 查看删除曲面后的图形。当选择菜单选项后，就可以删除该Isosurface曲面，得到的结果如图11.113所示。


图 11.113　删除曲面后的图形对象

step 12 修改Slice截面的颜色属性。选择图形界面中的"AllSlices" ⇨ "Color Faceted"选项，设置Slice界面的颜色属性，得到的结果如图11.114所示。


图 11.114　修改 Slice 截面的颜色属性

step 13 在Slice截面上添加等高线。选择Slice图形，然后单击鼠标右键，在弹出的现场菜单中选

择“Add Contour”⇨“Auto（Slice）”选项，如图 11.115 所示。

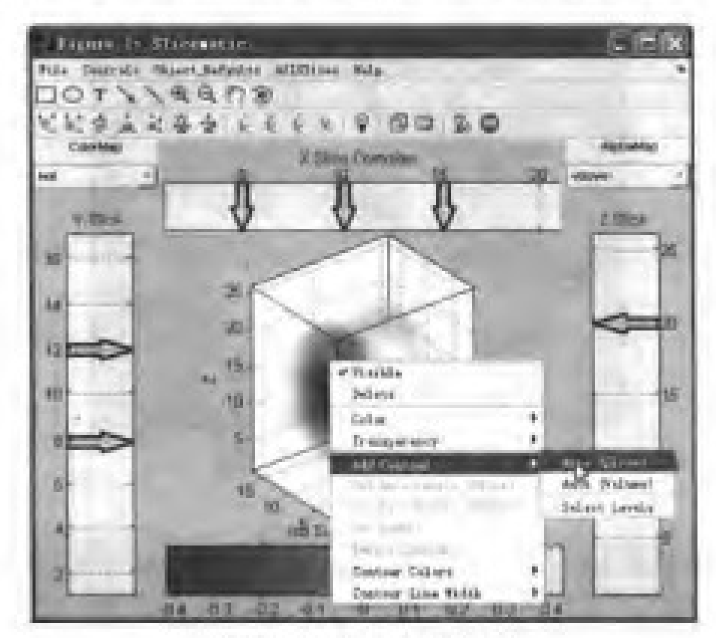


图 11.115 添加等高线



为了在后面的步骤中更好地显示等高线，将上面步骤中修改的颜色属性改为默认的属性，这样才能正常显示等高线。

step 14 查看图形结果。当选择上面的菜单选项后，得到的结果如图 11.116 所示。

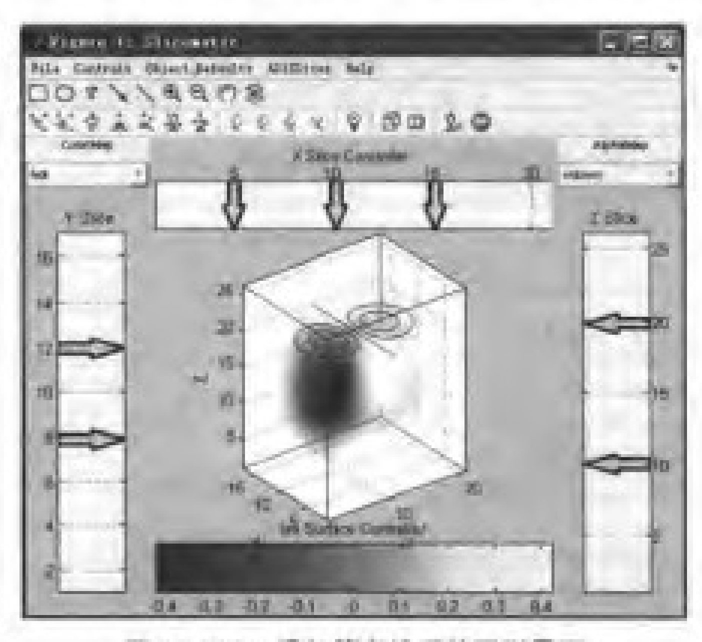


图 11.116 添加等高线后的图形界面

step 15 删除等高线。选择 Slice 图形，然后单击鼠标右键，在弹出的菜单中选择“Remove Contour”⇨“Auto（Slice）”选项，删除上面步骤中添加的等高线，如图 11.117 所示。

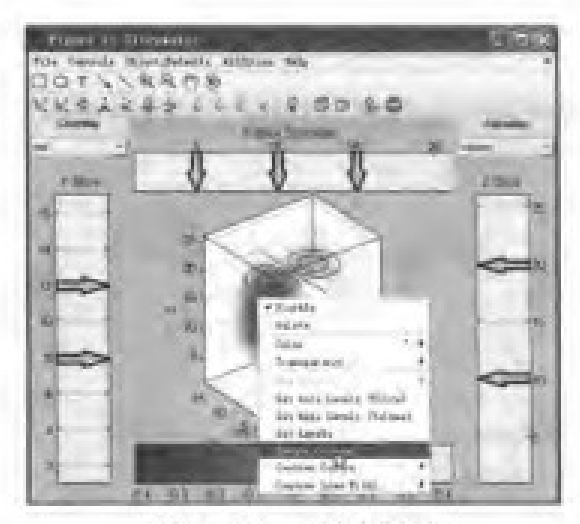


图 11.117 删除等高线



上面的步骤已经基本演示了该GUI对象的各种功能，关于该GUI其他具体的功能，在这里就不重复介绍了。

step 16 单击命令窗口工具栏中的 按钮，或者选择编辑栏中的 "File" ⇨ "New" ⇨ "M-file" 命令，打开一个空白的 M 文件编辑器，然后在 M 文件编辑器中输入下面的代码：

```
function v=blinnblob(centers,nx,ny,nz)
% Blinn's blobs
% centers=[ 20+8*cos(theta);20+8*sin(theta);20+zeros(1,6)]'
% v=blob(centers,40,40,40)
% isosurface(v,.125)
% From ACM Transactions on Graphics, July 1982, Volume 1, Number 3.
% "A Generalization of A;gebraic Surface Drawing" James F. Blinn
    x=makeXMat(nx,ny,nz);
    y=makeYMat(nx,ny,nz);
    z=makeZMat(nx,ny,nz);
    a=.05;
    b=1;
    numCenters=size(centers,1);
    v=zeros(nx,ny,nz);
    for i=1:numCenters
        dx=centers(i,1)-x;
        dy=centers(i,2)-y;
        dz=centers(i,3)-z;
        v=v+b*exp(-a*(dx.^2 + dy.^2 + dz.^2));
    end
function x=makeXMat(nx,ny,nz)
    x=repmat([ 1:ny] ,[ nx 1 nz] );
function y=makeYMat(nx,ny,nz)
    y=repmat([ 1:nx] ',[ 1 ny nz] );
function z=makeZMat(nx,ny,nz)
    z=repmat([ 1:nz] ,nx*ny,1);
z=reshape(z,[ nx ny nz] );
```

上面程序代码的主要功能是编写产生三维数据的函数blinnblob，该函数可以产生绘制图形的多维数据。最后，将上面的程序代码保存为 "blinnblob.m" 文件。

step 17 单击命令窗口工具栏中的□按钮，或者选择编辑栏中的"File"⇨"New"⇨"M-file"命令，打开一个空白的M文件编辑器，然后在M文件编辑器中输入下面的代码：

```
function slicebucky
% SLICEBUCKY - Make a bucky ball approximation for sliceomatic
  disp('Creating bucky ball approximation...');
  [b,v]=bucky;
  centers=32+20*v;
  v=blinnblob(centers,64,64,64);
  disp('Starting Sliceomatic');
  sliceomatic(v)
  daspect([1 1 1]);
  xlim([1 64]);
  ylim([1 64]);
  zlim([1 64]);
```

上面程序代码的主要功能是根据blinnblob函数产生的三维数据，调用前面步骤中编写的sliceomatic函数绘制Slice截面。最后，将上面的程序代码保存为"slicebucky.m"文件。

step 18 在MATLAB的命令窗口中输入如下代码：

```
>> slicebucky
系统会显示出下面的信息
Creating bucky ball approximation...
Starting Sliceomatic
Smoothing for IsoNormals...
Generating reduction volume...
```

step 19 查看图形结果。输入上面的程序代码后，按"Enter"键，在其中选择截面的坐标，如图11.118所示。

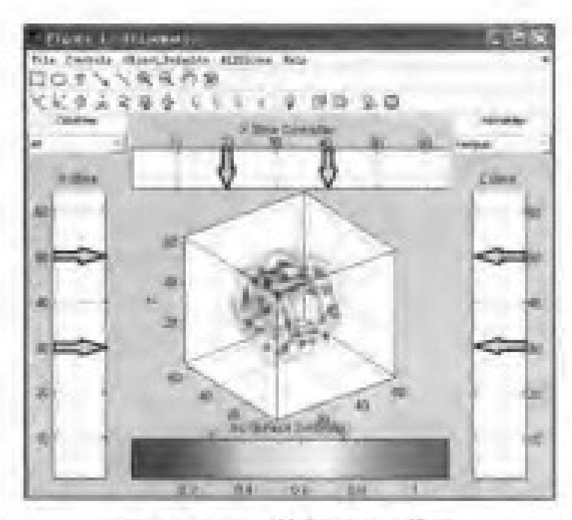


图 11.118　绘制Slice截面

step 20 修改Isosurface曲面的数值。在"Iso Surface Controller"控件中选择对应的数值，添加Isosurface曲面，得到的结果如图11.119所示。



可以采用和前面步骤类似的方法，设置该GUI对象的其他图形对象，并设置相应的图形属性，这里就不重复介绍了。

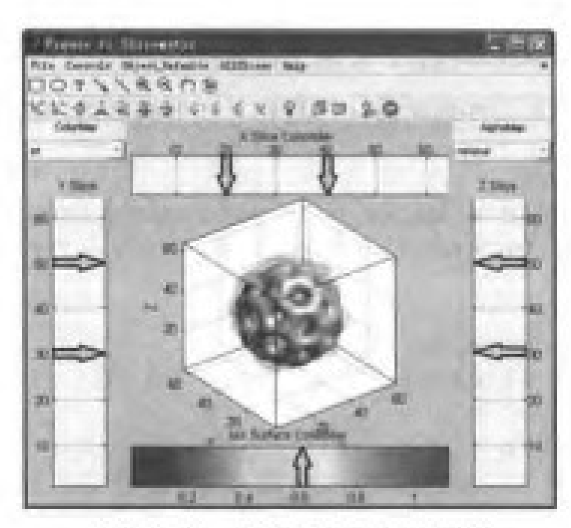


图 11.119　添加 Isosurface 曲面

11.10 小结

在本章中，介绍了如何使用M文件和GUIDE来创建GUI对象，如何添加GUI的图形控件，如何创建图形界面的各种菜单对象等。这些内容都是GUI对象的重要组成部分，内容也相对比较复杂，希望读者能够仔细体会。在后面的章节中，将主要介绍文件输入和输出的内容。

第12章 文 件 I/O

本章包括

- 打开外部文件
- 关闭外部文件
- 读取二进制文件
- 写入二进制文件
- 读取文本文件
- 写入文本文件
- 处理图像对象

为了加强 MATLAB 的应用功能，实现 MATLAB 和其他格式的文件的相互交换是很重要的内容。MATLAB 系统具有直接对磁盘文件进行访问的功能，不仅可以进行高层次的程序设计，也可以对低层次的文件进行读写操作，这样就增加了 MATLAB 程序设计的灵活性和兼容性。在 MATLAB 中，提供许多有关文件输入和输出的函数，使用这些函数可以很方便地实现各种格式的读取工作，大多数函数都是基于 C 语言的文件 I/O 函数，因此比较容易上手，下面简要介绍。

处理文件名称

为了实现各种不同格式文件的读取工作，MATLAB 首先提供能够处理文件路径或者名称的函数，使用这些函数，用户可以对文件路径进行各种处理：分割路径名称、组合路径名称等，下面使用简单的例子来说明如何使用这些函数。

例 12.1 获取文件路径的各种信息，并对路径各部分进行处理。

step 1 编写处理路径的代码。选择编辑栏中的 "File" ⇨ "New" ⇨ "M-file" 命令，打开 M 文件编辑器，然后在其中添加下面的程序代码：

```
function fileinfo(file)
%获取文件路径的各种信息
[pathstr,name,ext,versn] = fileparts(file);
%处理文件名称
if name=='fileinfo'
disp('This is the target file')
else
disp('This is not the target file')
end

%处理文件的后缀
if ext==strcat('.','xls')
disp('This is a excel file')
else
disp('This is not a excel file')
end
```

将上面的程序代码保存为 "fileinto.m" 文件，在后面的程序代码中将需要调用该函数来处理文件的名称和后缀信息。

step 2 处理文件路径信息。在 MATLAB 的命令窗口中输入下面的代码：

```
>> file = '\home\smartchen\matlab\filenifo.xls';
>> fileinfo(file)
This is not the target file
This is a excel file
>> file = '\home\smartchen\matlab\fileinfo.xls';
>> fileinfo(file)
This is the target file
This is a excel file
>> file = '\home\smartchen\matlab\fileinfo.txt';
>> fileinfo(file)
This is the target file
This is not a excel file
```

从上面的结果可以看出，在MATLAB中可以使用fileparts函数来返回文件路径各部分的信息，其完整的调用格式为：

```
[pathstr,name,ext,versn] = fileparts('filename')
```

在该函数返回的参数中，pathstr 表示的是文件路径，name 则是文件名称，ext 返回的则是文件的后缀（包含后缀前面的点号），versn 返回的是文件版本。



考虑到用户使用的操作系统的问题，MATLAB 提供 "filesep" 函数来返回文件路径的分隔符。在 Windows 操作系统中，该函数返回的是 "\"；如果在 UNIX 操作系统中，该函数返回的是 "/"。

例 12.2 利用路径各部分的内容创建完整的文件路径。

step 1 创建文件路径。在 MATLAB 的命令窗口中输入下面的代码：

```
>> file = '\home\smartchen\matlab\filenifo.xls';
>> [pathstr,name,ext,versn] = fileparts(file);
>> filecontr=fullfile(pathstr,[name ext versn]);
```

step 2 查看程序结果。上面的程序代码可以得到下面的结果：

```
>> pathstr
pathstr =
\home\smartchen\matlab
>> name
name =
filenifo
>> ext
ext =
.xls
>> filecontr
filecontr =
\home\smartchen\matlab\filenifo.xls
```

从上面的结果可以看出，在 MATLAB 中可以使用 fullfile 命令来得到完整的文件路径，其完整的调用格式如下：

```
f = fullfile('dir1','dir2',...,'filename')
```

在上面的命令中，前面的参数分别表示的是文件的路径，最后一个参数表示的是文件名称，该文件名称中如果不包含后缀，则创建的完整路径也不包含后缀。



除了上面介绍的函数之外，MATLAB还提供tempdir和tempname两个命令，用户可以使用这些命令来了解临时路径的信息，具体的使用方法请查看对应的帮助文件。

12.2 打开和关闭文件

在对文件进行处理的所有工作当中，打开文件或者关闭文件都是十分基础的操作。在本小节中，将介绍如何在MATLAB中打开和关闭文件。根据操作系统的要求，在程序代码中需要使用或者创建某个磁盘文件的时候，必须向操作系统发出打开文件的命令；使用完毕之后，也必须向操作系统发出关闭文件的命令。

对于和MATLAB同等层次的文件，可以使用 load、save 等命令对该文件进行操作，具体的操作方法请参考第 1 章的内容。在本小节中，将主要介绍如何在 MATLAB 中读取低层次的数据文件方法。

12.2.1 打开文件

在本小节中，将主要使用简单的实例介绍如何在MATLAB中打开磁盘中的文件，然后详细介绍对应命令的使用方法。

例 12.3 在 MATLAB 中使用 fopen 命令打开磁盘文件。

step 1 以读写的方式打开磁盘文件 fgetl.m。在 MATLAB 的命令窗口中输入下面的代码：

```
>> [ fid,message] = fopen('fgetl.m','r+')
```

step 2 查看程序结果。在输入上面的代码后，得到的结果如下：

```
fid =
    -1
message =
No such file or directory;
```

由于在运行该命令的时候，命令查找的范围是用户设置的文件路径为：\MATLAB7.0\work，而 fgetl.m 是函数 fgetl 的 M 文件，默认保存路径为 MATLAB7.0\toolbox\matlab\iofun。

step 3 查看 M 文件的程序代码。为了和后面步骤中打开的文件内容相比较，下面列出该文件的代码：

```
function tline = fgetl(fid)
try
    [ tline,lt] = fgets(fid);
    tline = tline(1:end-length(lt));
    if isempty(tline)
        tline = '';
```

```
    end

catch
    if nargin ~= 1
        error (nargchk(1,1,nargin,'struct'))
    end
    if isempty(fopen(fid))
        error ('MATLAB:fgetl:InvalidFID','Invalid file identifier.')
    end
    rethrow(lasterror)
end
```



尽管在系统中存在该函数文件，但是该文件不在搜索路径上，当 fopen 以读写方法打开的时候，将会返回错误信息。

step 4 以只写的方式打开磁盘文件 fgetl.m。在 MATLAB 的命令窗口中输入下面的代码：

```
>> [ fid,message] = fopen('fgetl.m','w')
```

step 5 查看程序结果。上面的程序代码得到的结果如下：

```
fid =
     4
message =
     ''
```

前面步骤已经提到，fgetl.m 并不在命令搜索路径上，但是该命令并没有返回错误信息，而是返回了正整数的信息，表示已经打开该文件。这是因为当以只写方式打开文件时，如果命令没有搜索到对应的文件，则会自动创建该文件。因此，当用户使用该命令后，系统会在搜索路径 \MATLAB7.0\work 上创建一个空白的 M 文件，该文件名称为 fgetl。



在 MATLAB 中，fopen 命令打开文件的方式是只读方式，因此在默认情况下，如果用户不输入打开文件的方式，MATLAB 就会使用只读方式打开文件。

step 6 以读写的方式打开磁盘文件 fgetl.m。在 MATLAB 的命令窗口中输入下面的代码：

```
>> [ fid,message] = fopen('fgetl.m','r+')
```

step 7 查看程序代码的结果。上面的程序代码可以得到下面的结果：

```
fid =
     4
message =
     ''
```



从上面的结果可以看出，当用户首先以只写的方式打开磁盘文件 fgetl.m 后，命令会自动创建对应的空白文件；再次使用读写方式打开该文件时，系统会打开对应的空白文件，并返回打开文件的数值信息。

上面例子基本演示了MATLAB中fopen命令的使用方法，其对应的完整调用格式如下：

- [fid,message] = fopen(filename, mode)
- [fid,message] = fopen(filename, mode, machineformat)

在上面的命令中，filename表示的是打开文件的名称，mode表示打开文件的方式，其具体的类型如下。

- "r"：以只读方式打开文件。
- "w"：以只写方式打开文件，并覆盖原来的内容。
- "a"：增补文件，在文件尾部增加数据。
- "r+"：读写文件。
- "w+"：创建一个新文件或者删除已有的文件内容，并进行读写操作。
- "a+"：读取和增补文件。



在默认情况下，MATLAB会选择使用二进制的方式打开文件，而在该方式下，字符串不会被特殊处理。如果用户需要用文本形式打开文件，则需要在上面的mode字符串后面添加"t"，例如，"rt"、"rt+"等。

在上面两个命令中，fid是一个非负整数，一般被称为文件标识。在MATLAB中，用户对文件的任何操作，都需要通过fid参数来传递，MATLAB会根据fid的数值来标识所有已经打开的文件，然后实现对文件的读、写和关闭等各种操作。如果程序代码得到fid的数值是 -1，则表示fopen不能打开对应的文件，可能是因为该文件本身不存在，用户却以读写的方式来打开，或者文件存在但是不在搜索路径上。



具体的fid数值是没有太大的意义的，一般是由系统自动获取的，必须仔细查看或者打开进行处理，只需查看fid数值是否为 -1 即可。

12.2.2 关闭文件

在前面小节中曾经介绍过，在打开文件后，如果完成了对应的读写工作，应该关闭文件；否则打开的文件就会过多，造成系统资源的浪费。在本小节中将以一个简单的实例来说明如何在MATLAB中关闭对应的文件。

例 12.4 在MATLAB中关闭对应的磁盘文件。

step 1 创建文件fgetl.m，然后删除该文件。在MATLAB的命令窗口中输入下面的代码：

```
>> [ fid,message] = fopen('fgetl.m','w');
>> delete fgetl.m
```

step 2 查看程序代码的结果。上面的程序代码中可以得到下面的结果：

```
Warning: File not found or permission denied.
```



上面的结果表明，当用户使用fopen创建对应的空白文件fgetl.m，并打开对应的文件后，如果在关闭该文件前，直接删除文件，系统会提示用户删除命令被拒绝。

step 3 首先关闭文件，然后删除该文件。在MATLAB的命令窗口中输入下面的代码：

```
>> status=fclose(fid);
>>delete fgetl.m;
>> [ fid,message] = fopen('fgetl.m','r+');
```

step 4 查看程序代码的结果。当用户输入上面的程序代码后，得到的结果如下：

```
fid =
    -1
message =
No such file or directory
```



从上面的结果可以看出，当用户首先关闭对应的文件，再删除创建的文件后，最后再次打开对应的文件，返回的信息是无法找到文件，表明已经删除了该文件。

上面的例子演示了如何在MATLAB中关闭文件。在MATLAB中，用户可以使用fclose命令关闭已经打开的文件，其具体的调用命令如下：

```
status = fclose(fid)
status = fclose('all')
```

在上面的命令中，fid表示使用fopen命令得到的文件标识参数，第二个命令表示使用命令删除所有已经打开的文件。如果使用该命令得到的结果status=0，则表示关闭文件的操作成功，否则得到的结果status=-1。



从上面两个小节中可以看出，在MATLAB中打开和关闭文件对应的命令比较简单，但是笔者还是不建议在程序代码中循环使用这些命令，因为这样会降低程序的效率。

12.3 处理二进制文件

对于MATLAB而言，二进制文件是相对比较容易处理的。和后面要介绍的文本文件或者XML文件相比较，这些文件是比较容易和MATLAB进行交互的。因此，在本小节中将首先介绍如何读取和写入二进制文件。

12.3.1 读取M文件

常见的二进制文件包括.m、.dat等，可以使用MATLAB中提供的fread来读取对应的文件。本小节还是以具体的实例来说明如何使用该命令。

例 12.5 在MATLAB中读取ball.m文件的内容。

step 1 查看ball.m文件的内容。为了检查在后面的步骤中使用fread命令的结果，本步骤首先使用

M文件编辑器查看该文件的代码内容，得到结果如下：

```
% Constants
conv=pi/180;
grav=-9.82;
vo=input('Enter the initial velocity:');
range=zeros(1,91);
% 计算最长的水平距离
for ii=1:91
      theta=ii-1;
      vxo=vo*cos(theta*conv);
      vyo=vo*sin(theta*conv);
      max_time=-2*vyo/grav;
      range(ii)=vxo*max_time;
end
% 读入水平距离的列表
fprintf('Range versus angle theta:\n');
for ii=1:5:91
     theta=ii-1;
     fprintf('%2d %8.4f\n',theta,range(ii));
end
% 计算最大的角度和水平距离
[maxrange index]=max(range);
maxangle=index-1;
fprintf('\n Max range is %8.4f at %2d degrees.\n',maxrange,maxangle);
% 绘制轨迹
for ii=5:10:80
      theta=ii;
      vxo=vo*cos(theta*conv);
      vyo=vo*sin(theta*conv);
      max_time=-2*vyo/grav;
      % 计算坐标数值x, y
      x=zeros(1,21);
      y=zeros(1,21);
      for jj=1:21
          time=(jj-1)*max_time/20;
          x(jj)=vxo*time;
          y(jj)=vyo*time+0.5*grav*time^2;
      end
      plot(x,y,'g');
      if ii==5
          hold on;
      end
end
% 添加标题和坐标轴名称
title('\bf Trajectory of Ball vs Initial Angle\theta');
xlabel('\bf\itx \rm\bf(meters)');
ylabel('\bf\ity \rm\bf(meters)');
axis([ 0 max(range)+5 0 -vo^2/2/grav]);
grid on;
% 绘制最长距离的轨迹曲线
vxo=vo*cos(maxangle*conv);
vyo=vo*sin(maxangle*conv);
max_time=-2*vyo/grav;
      % 计算坐标轴数值x, y
      x=zeros(1,21);
```

```
y=zeros(1,21);
for jj=1:21
    time=(jj-1)*max_time/20;
    x(jj)=vxo*time;
    y(jj)=vyo*time+0.5*grav*time^2;
end
plot(x,y,'r','Linewidth',2);
hold off
```



之所以在本步骤中打开该M文件，是因为该M文件包含MATLAB中的各种常见命令，利用该文件查看fread命令对各种代码的处理方法。

step 2 读取ball.m文件的内容。在MATLAB的命令窗口中输入下面的代码：

```
>> [fid,message]=fopen('ball.m','r+');
>> data=fread(fid);
```

当用户读取上面的文件后，可以查看data变量的属性：

```
>> whos data
  Name        Size                       Bytes  Class
  data     2522x1                        20176  double array

Grand total is 2522 elements using 20176 bytes
```



从上面的结果中可以看出，data是数值数组，也就是说，尽管用户打开的文件中是程序代码，但是使用fread读取该文件后，得到的是数值数组。

step 3 查看读取的数据结果。在MATLAB的命令窗口中输入下面的代码：

```
>> data
```

step 4 查看程序代码的结果。在命令窗口中输入变量名称，得到的结果如下：

```
data =
    37
    32
    83
    99
  ............//省略了部分数据
   111
   102
   102
    13
    10
    13
    10
    13
    10
```



根据前面步骤的结果，数值数组data的大小是2522x1，因此上面的结果中省略了部分数据，列出的是部分数据结果。

step 5 查看程序代码。在命令窗口中使用下面的命令代码：

```
>> disp(char(data'))
上面的程序代码得到的结果如下：
%  Constants
conv=pi/180;
grav=-9.82;
vo=input('Enter the initial velocity:');
range=zeros(1,91);
% Calculate maximum ranges
for ii=1:91
     theta=ii-1;
     vxo=vo*cos(theta*conv);
     vyo=vo*sin(theta*conv);
     max_time=-2*vyo/grav;
     range(ii)=vxo*max_time;
end
.......................................................//省略了部分代码
          time=(jj-1)*max_time/20;
        x(jj)=vxo*time;
        y(jj)=vyo*time+0.5*grav*time^2;
     end
     plot(x,y,'r','Linewidth',2);
     hold off
```



从结果的角度来看，上面的命令代码和 "type ball.m" 是相同的，相当于将该文件中的所有代码都显示出来。

12.3.2 读取 TXT 文件

TXT 文件也是比较常见的二进制文件。在本小节中，将以一个简单的例子来介绍如何在 MATLAB 中读取 TXT 文件。

例 12.6 在 MATLAB 中读取 readtxt.txt 文件中的内容，并对读取的内容进行处理。

step 1 在本实例中，readtxt.txt 是文本文件，其中的内容包含下面的字母文字：

```
This is a txt file
```

step 2 读取 readtxt.txt 文件的内容。在 MATLAB 的命令窗口中输入下面的代码：

```
>> fid = fopen('readtxt.txt', 'r');
>> data=fread(fid);
```

step 3 查看程序代码的结果。在命令窗口中输入变量名称，得到的结果如下：

```
>> data
data =
    84
   104
   105
```

```
  115
   32
  105
  115
   32
   97
...................//省略了部分内容
   32
  102
  105
  108
  101
   13
   10
```



和前面步骤类似，默认情况下读取文件得到的数组是数值数组，而不是读取文件里的文本内容，需要使用其他具体命令来转换。

step 4 读取 readtxt.txt 文件的部分内容。在 MATLAB 的命令窗口中输入下面的代码：

```
>> fid = fopen('readtxt.txt', 'r');
c = fread(fid, 5)
```

step 5 查看程序代码的结果。在命令窗口中输入变量名称，得到的结果如下：

```
c =
    84
   104
   105
   115
    32
```



上面的结果可以看出，可以使用 fread 命令的第二个参数来设定读取文件的部分内容，在上面的步骤中选择的是 5，表示的是选择前面 5 个数值内容。

step 6 读取 readtxt.txt 文件的部分文本内容。在 MATLAB 的命令窗口中输入下面的代码：

```
>> fid = fopen('readtxt.txt', 'r');
>> c = fread(fid, '*char');
>> sprintf(c)
```

step 7 查看程序代码的结果。输入代码后，按 "Enter" 键，得到的结果如下：

```
ans =

This is a txt file
```

step 8 处理 readtxt.txt 文件的部分文本内容。在 MATLAB 的命令窗口中输入下面的代码：

```
>> fid = fopen('readtxt.txt', 'r');
>>c1 = fread(fid, 5, '*char');
```

```
>>c2 = fread(fid, 3, '*char');
>>c3 = fread(fid, 2, '*char');
>> c4 = fread(fid, 4, '*char');
>> c5 = fread(fid, 5, '*char');
>> sprintf('%c', c1, ' * ', c2, ' * ', c3,' * ',c4,' * ',c5)
```

step 9 查看程序代码的结果。输入代码后，按“Enter”键，得到的结果如下：

```
ans =

This  * is  * a  * txt  * file
```

上面的例子并不复杂，基本上演示了如何在MATLAB中读取二进制文件的方法。在MATLAB中，读取二进制文件的命令是fread，其调用格式如下：

A = fread(fid, count, precision)

在上面的命令中，参数fid表示使用fopen命令打开的文件标识，参数count表示读取二进制文件的大小，可以选取下面三个参数。

- n：读取前面n个整数，并写入一个向量中。
- inf：读取文件，直到文件结尾处。
- [m,n]：读取数据到m*n的矩阵中，按照列排列，n可以是inf，但是m不能是inf。

最后一个参数precision用来控制二进制数据转换成为MATLAB矩阵时的精度，具体的精度参数请用户查看相应的帮助文件。

写入二进制文件

在MATLAB中，如果用户希望按照指定的二进制文件格式将矩阵的元素读入文件中，可以使用fwrite命令来完成这样的任务。和前面小节类似，在本小节中将使用简单的例子来说明如何使用该命令。

例 12.7 在MATLAB中使用fwrite命令来写入二进制文件。

step 1 在MATLAB的命令窗口中输入下面的程序代码：

```
>>fid = fopen('magic5.txt','wb');
>>fwrite(fid,magic(5),'int32');
>>fid=fopen('magic5.bin','r');
>>data=fread(fid,[ 5,5] ,'int32');
>>A=data';
```

step 2 查看程序代码的结果。在命令窗口中输入变量名称，得到的结果如下：

```
A =
    17    23     4    10    11
    24     5     6    12    18
     1     7    13    19    25
     8    14    20    21     2
    15    16    22     3     9
>> magic(5)
ans =
    17    24     1     8    15
```

```
    23      5      7     14     16
     4      6     13     20     22
    10     12     19     21      3
    11     18     25      2      9
```



从上面的结果可以看出，首先创建并打开一个空白的二进制文件，然后将MATLAB中创建的Magic矩阵读入该文件后，再打开该文件，阅读该文件的内容。

12.4 处理文本文件

为了处理文本文件，MATLAB提供多种处理函数，使用不同的格式读取不同数据类型的文本文件。在本小节中，将通过实例来详细介绍。

12.4.1 读取文本文件

在MATLAB中，提供多个函数来读取文本文件中的数据，其中比较常见的包括函数csvread、dlmread和textread等，这些函数在实际使用中有各自的使用范围和特点。在本小节中，将主要使用实例来说明这些函数的使用方法。

例 12.8 在MATLAB中使用csvread和dlmread命令来读取文本文件。

step 1 查看原始的数据文件。在本实例中，需要读取的文件是txtlist.dat，其文件中包含的主要数据如下：

```
02, 04, 06, 08, 10, 12
03, 06, 09, 12, 15, 18
05, 10, 15, 20, 25, 30
07, 14, 21, 28, 35, 42
11, 22, 33, 44, 55, 66
```

step 2 使用命令读取该数据文件。在MATLAB的命令窗口中输入下面的程序代码：

```
>> m1 = csvread('txtlist.dat');
>> m2 = csvread('txtlist.dat', 2, 0);
>>m3= csvread('txtlist.dat', 2, 0, [2,0,3,3]);
>>m4=dlmread('txtlist.dat');
```

step 3 查看程序代码的结果。在命令窗口中输入变量名称，得到的结果如下：

```
m1 =
     2      4      6      8     10     12
     3      6      9     12     15     18
     5     10     15     20     25     30
     7     14     21     28     35     42
    11     22     33     44     55     66
m2 =
     5     10     15     20     25     30
     7     14     21     28     35     42
    11     22     33     44     55     66
m3 =
```

```
     5    10    15    20
     7    14    21    28
m4 =
     2     4     6     8    10    12
     3     6     9    12    15    18
     5    10    15    20    25    30
     7    14    21    28    35    42
    11    22    33    44    55    66
```



从上面的结果可以看出，使用上面两个命令，可以根据不同的条件来读取原始文本文件中的数据。其中在dlmread命令中，用户可以自行设置数据之间的分隔符。

例 12.9 在 MATLAB 中使用 textread 命令来读取文本文件。

step 1 查看原始的数据文件。在本实例中，需要读取的文件是 txtlist2.txt，其文件中包含的第一行数据如下：

```
Sally    Level1 12.34 45 Yes
```

step 2 使用命令读取该数据文件。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>>[ names, types, x, y, answer] = textread('txtlist2.txt', ...
'%s %s %f %d %s', 1);
```

step 3 查看程序代码的结果。在命令窗口中输入变量名称，得到的结果如下：

```
names =
    'Sally'
types =
    'Level1'
x =
   12.3400
y =
    45
answer =
    'Yes'
```



从上面的结果可以看出，在textread命令中，可以指定读取数据的格式，得到对应的输入结果，具体的使用方法将在后面内容中介绍。

例 12.10 在 MATLAB 中使用 txtscan 命令来读取文本文件。

step 1 查看原始的数据文件。在本实例中，需要读取的文件是 txtscan.dat，其文件中包含的数据如下：

```
Sally  Level1 12.34 45 1.23e10 inf NaN Yes
Joe    Level2 23.54 60 9e19 -inf 0.001 No
Bill   Level3 34.90 12 2e5 10 100 No
```

step 2 使用命令读取该数据文件。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> fid = fopen('txtscan.dat');
```

```
>> C1 = txtscan(fid, '%s %s %f32 %d8 %u %f %f %s');
>> fclose(fid);
```

step 3 查看程序代码的结果。在命令窗口中输入变量名称，得到的结果如下：

```
>> whos C1
  Name      Size                    Bytes  Class
  C1        1x8                      1169  cell array

Grand total is 69 elements using 1169 bytes
```



可以看出，使用txtscan命令得到的结果将会被存储在元胞数组中，该元胞数组包含的列数就是原始数据文件中使用分隔符隔开的数据列。

step 4 查看C1数组的结果。在命令窗口中输入下面的程序代码：

```
for i=1:8
disp(C1{i}');
end
```

step 5 查看程序代码的结果。输入代码后，按"Enter"键，得到的结果如下：

```
'Sally'    'Joe'    'Bill'
    'Level1'    'Level2'    'Level3'
   12.3400   23.5400   34.9000
   45   60   12
  4294967295  4294967295      200000
   Inf  -Inf    10
       NaN    0.0010  100.0000
    'Yes'    'No'    'No'
```

为了方便用户在后面的步骤中比较各种查看方法的不同，将上面的结果整理如下：

```
C1{1}  = {'Sally'; 'Joe'; 'Bill'}            class cell
C1{2}  = {'Level1'; 'Level2'; 'Level3'}      class cell
C1{3}  = [12.34; 23.54; 34.90]               class single
C1{4}  = [45; 60; 12]                        class int8
C1{5}  = [1.23e10; 9e19; 2e5]                class uint32
C1{6}  = [Inf; -Inf; 10]                     class double
C1{7}  = [NaN; 0.001; 100]                   class double
C1{8}  = {'Yes'; 'No'; 'No'}                 class cell
```

step 6 读取原始文件，并忽略第三列数据。在命令窗口中输入下面的程序代码：

```
>> fid = fopen('txtscan.dat');
>>C2 = txtscan(fid, '%7c %6s %*f %d8 %u %f %f %s');
>>fclose(fid);
```

step 7 查看C2的属性。在命令窗口中输入下面的程序代码：

```
>> whos C2
  Name      Size                    Bytes  Class
  C2        1x7                       935  cell array
```

```
Grand total is 71 elements using 935 bytes
```

step 8 查看C2数组的结果。在命令窗口中输入下面的程序代码：

```
for i=1:7
disp(C2{i}');
end
```

step 9 查看程序代码的结果。输入代码后，按"Enter"键，得到的结果如下：

```
'Sally'    'Joe'    'Bill'
    'Level1'    'Level2'    'Level3'
   45    60    12
  4294967295  4294967295      200000
   Inf  -Inf     10
       NaN    0.0010   100.0000
    'Yes'    'No'    'No'
```

同样，将上面的结果整理如下：

```
C2{1}  = ['Sally  '; 'Joe    '; 'Bill   ']     class char
C2{2}  = {'Level1'; 'Level2'; 'Level3'}        class cell
C2{3}  = [45; 60; 12]                          class int8
C2{4}  = [1.23e10; 9e19; 2e5]                  class uint32
C2{5}  = [Inf; -Inf; 10]                       class double
C2{6}  = [NaN; 0.001; 100]                     class double
C2{7}  = {'Yes'; 'No'; 'No'}                   class cell
```



从上面的结果可以看出，当在txtscan命令中使用textscan(fid, '%7c %6s %*f %d8 %u %f %f %s')后，其中%*f所替代的对应数据列会被跳过，不被读入。

step 10 仅仅读取原始文件的第一列数据。在命令窗口中输入下面的程序代码：

```
>> fid = fopen('txtscan.dat');
names = textscan(fid, '%s%*[^\n]');
fclose(fid);
```

step 11 查看names的属性。在命令窗口中输入下面的程序代码：

```
>> whos names
  Name        Size                    Bytes  Class
  names       1x1                       264  cell array

Grand total is 16 elements using 264 bytes
```

step 12 查看程序代码的结果。在命令窗口中输入下面的程序代码：

```
>> B= names{1}'
B =
'Sally'    'Joe'    'Bill'
```

step 13 清除原始第三列数据前面的标签。在命令窗口中输入下面的程序代码：

```
>> fid = fopen('txtscan.dat');
>>C3 = textscan(fid, '%s Level%u8 %f32 %d8 %u %f %f %s');
>>fclose(fid);
>> whos C3
```

step 14 查看程序代码的结果。输入代码后，按“Enter”键，得到的结果如下：

```
  Name      Size                    Bytes  Class
  C3        1x8                       956  cell array

Grand total is 51 elements using 956 bytes
```

step 15 查看C2的属性。在命令窗口中输入下面的程序代码：

```
>> for i=1:8
disp(C3{i}');
end
```

step 16 查看程序代码的结果。输入代码后，按“Enter”键，得到的结果如下：

```
'Sally'    'Joe'    'Bill'
    1     2     3
   12.3400    23.5400    34.9000
   45    60    12
  4294967295  4294967295       200000
   Inf   -Inf     10
       NaN     0.0010  100.0000
    'Yes'    'No'    'No'
```



从上面的结果中可以看出，相对于原始的第二列数据，通过该命令得到的第二数据清除了字符串“Level”，只留下了数值代码。

最后，将上面的结果整理如下：

```
C3{1}  = {'Sally'; 'Joe'; 'Bill'}       class cell
C3{2}  = [1; 2; 3]                      class uint8
C3{3}  = [12.34; 23.54; 34.90]          class single
C3{4}  = [45; 60; 12]                   class int8
C3{5}  = [1.23e10; 9e19; 2e5]           class uint32
C3{6}  = [Inf; -Inf; 10]                class double
C3{7}  = [NaN; 0.001; 100]              class double
C3{8}  = {'Yes'; 'No'; 'No'}            class cell
```

根据上面的例子可以看出，在MATLAB中读取文本文件的命令如下。

- M = csvread('filename', row, col)：在该命令中，filename是需要打开的文本文件，row是需要读取的行，col则是需要读取的数据列。
- A = dlmread('filename', delimiter)：在该命令中，filename是需要打开的文本文件，delimiter表示用户自行指定的分隔符。
- C = textscan(fid, 'format', N)：在该命令中，fid是使用fopen命令得到的文件标识fid，format则表示读取文件的变量格式，N表示读取数据的循环次数。

12.4.2 使用csvwrite命令读入文本文件

上面介绍过如何在MATLAB中读取文本文件。在本小节中，将利用一些简单的实例来详细介绍如何在MATLAB中写入文本文件。

例 12.11 使用csvwrite命令向文本文件写入MATLAB的数据。

step 1 将数据写入csvlist文本文件中。在MATLAB的命令窗口中输入下面的程序代码：

```
m = [ 3 6 9 12 15; 5 10 15 20 25; 7 14 21 28 35; 11 22 33 44 55];
csvwrite('csvlist.dat',m)
type csvlist.dat
```

step 2 查看程序代码的结果。输入代码后，按"Enter"键，得到的结果如下：

```
3,6,9,12,15
5,10,15,20,25
7,14,21,28,35
11,22,33,44,55
```

step 3 将数据写入csvlist文本文件中，并在数据列的前侧添加两个数据列。在命令窗口输入下面的代码：

```
csvwrite('csvlist.dat',m,0,2)
type csvlist.dat
```

step 4 查看程序代码的结果。输入代码后，按"Enter"键，得到的结果如下：

```
,,3,6,9,12,15
,,5,10,15,20,25
,,7,14,21,28,35
,,11,22,33,44,55
```

step 5 将数据写入csvlist文本文件中，并在数据列的前侧添加4个数据列，同时在数据列上方添加2个数据行。在命令窗口输入下面的代码：

```
csvwrite('csvlist.dat',m,2,4)
type csvlist.dat
```

step 6 查看程序代码的结果。输入代码后，按"Enter"键，得到的结果如下：

```
,,,,,,,,
,,,,,,,,
,,,,3,6,9,12,15
,,,,5,10,15,20,25
,,,,7,14,21,28,35
,,,,11,22,33,44,55
```



在上面的命令中，都是在没有对应数据文件之前使用写入命令，但是使用该命令会首先创建该文件，然后实现写入任务。



使用 dlmwrite 命令读入文本文件

在MATLAB中，还可以使用dlmwirte命令来读入文本文件，下面使用具体的实例来说明如何使用该命令。

例 12.12 使用 dlmwrite 命令向文本文件写入 MATLAB 的数据。

step 1 将数据写入 myfile 文本文件中。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> m=rand(6);
dlmwrite('myfile.txt', m, 'delimiter', '\t', 'precision', 5)
type myfile.txt
```

step 2 查看程序代码的结果。输入代码后，按 "Enter" 键，得到的结果如下：

```
0.84622  0.68128   0.30462   0.15087   0.49655   0.34197
0.52515  0.37948   0.18965   0.6979    0.89977   0.28973
0.20265  0.8318    0.19343   0.37837   0.82163   0.34119
0.67214  0.50281   0.68222   0.86001   0.64491   0.53408
0.83812  0.70947   0.30276   0.85366   0.81797   0.72711
0.01964  0.42889   0.54167   0.59356   0.66023   0.30929
```

step 3 修改数据精度，然后将数据写入 myfile 文本文件中。输入下面的程序代码：

```
>> m=rand(6);
dlmwrite('myfile.txt', m, 'delimiter', '\t', 'precision', 3)
type myfile.txt
```

step 1 查看程序代码的结果。输入代码后，按 "Enter" 键，得到的结果如下：

```
0.838    0.695    0.173    0.137    0.284    0.516
0.568    0.621    0.98     0.0118   0.469    0.334
0.37     0.795    0.271    0.894    0.0648   0.433
0.703    0.957    0.252    0.199    0.988    0.226
0.547    0.523    0.876    0.299    0.583    0.58
0.445    0.88     0.737    0.661    0.423    0.76
```

step 3 向 myfile 文本文件中写入多行数据。输入下面的程序代码：

```
>> M = ones(5);
dlmwrite('myfile.txt', [M*5 M/5], ' ')
dlmwrite('myfile.txt', eye(4),'-append', ...
   'roffset', 1, 'delimiter', ' ')
type myfile.txt
```

step 6 查看程序代码的结果。输入代码后，按 "Enter" 键，得到的结果如下：

```
5 5 5 5 5 0.2 0.2 0.2 0.2 0.2
5 5 5 5 5 0.2 0.2 0.2 0.2 0.2
5 5 5 5 5 0.2 0.2 0.2 0.2 0.2
5 5 5 5 5 0.2 0.2 0.2 0.2 0.2
5 5 5 5 5 0.2 0.2 0.2 0.2 0.2

1 0 0 0
0 1 0 0
```

```
0 0 1 0
0 0 0 1
```

从上面的例子中可以看出，在MATLAB中，可以使用csvread和dlmread来写入文本文件，它们的调用格式如下。

- csvwrite('filename',M,row,col)：在该命令中，filename表示的是数据写入的文件名称，M是对应的数据矩阵，row和col分别表示在原始数据基础上添加的数据行和数据列。
- dlmwrite('filename', M, '-append', attribute-value list)：在该命令中，filename表示数据写入的文件名称，M是对应的数据矩阵，'-append'表示保留原来的写入工作，最后的参数表示用户可以根据情况设置相应的参数属性，具体的属性列表请查看对应的帮助文件。

12.5 处理图像

在MATLAB中，图像一直是十分重要的组成部分，MATLAB可以在图像处理方面发挥多种作用，并完成各种复杂的工作。MATLAB还专门提供Image Acquisition、Image Processing等工具箱，完成各种复杂的图像处理工作。本节将以一个简单的实例来说明如何在MATLAB中读取和编辑图像对象。

例 12.13 向MATLAB中读入某图像文件，并进行适当的处理。

step 1 读入图像文件。输入下面的程序代码：

```
>> RGB = imread('gantrycrane.png');
imshow(RGB);
```

step 2 查看图形结果。输入代码后，按"Enter"键，得到的图形如图12.1所示。



图 12.1 读入图像文件

step 3 查看图像信息数组。在MATLAB命令窗口中输入下面的程序代码：

```
>> whos
  Name      Size                              Bytes  Class
  RGB      264400x3                          316800  uint8 array

Grand total is 316800 elements using 316800 bytes
```



在MATLAB中，imread函数将会返回图像的数据，并将数据保存在数组中，如果使用命令读取色图，数组将是三维数组，大小为 M*N*3。同时，该数组的数据类型取决于图像文件的数据类型。

step 4 在图像文件中添加文字信息。输入下面的程序代码：

```
%添加说明文字
text(size(RGB,2),size(RGB,1)+15,'Image courtesy of Jeff Mather',...
    'FontSize',7,'HorizontalAlignment','right');
%添加直线
line([ 300 328] ,[ 85 103] ,'color',[ 1 1 0] );
line([ 268 255] ,[ 85 140] ,'color',[ 1 1 0] );

%添加对应的说明文字
text(150,72,'Measure the angle between these beams','Color','y',...
    'FontWeight', 'bold');
```

step 5 查看图形结果。输入代码后，按"Enter"键，得到的图像结果如图 12.2 所示。



图 12.2 添加说明文字后的图像

step 6 选取需要测量角度的部分图像。在命令窗口中输入下面的程序代码：

```
% you can obtain the coordinates of the rectangular region using
% pixel information displayed by imview
start_row = 34;
start_col = 208;

cropRGB = RGB(start_row:163, start_col:400, :);

imshow(cropRGB)

% Store (X,Y) offsets for later use; subtract 1 so that each offset
will
% correspond to the last pixel before the region of interest
offsetX = start_col-1;
offsetY = start_row-1;
```

step 7 查看图形结果。输入代码后，按"Enter"键，得到的图像结果如图 12.3 所示。



在上面的程序代码中，起始参数设置的是选取图像对象的起始点的像素信息数据，如果用户希望了解这些信息的具体内容，可以使用 imview 函数查看具体信息。



图 12.3 选取部分图像

step 8 将上面的图像转换为黑白原色图像。在命令窗口中输入下面的程序代码：

```
% 进行图像数据转换
I = rgb2gray(cropRGB);
threshold = graythresh(I);
BW = im2bw(I,threshold);
BW = ~BW;  % complement the image (objects of interest must be white)
imshow(BW)
```

step 9 查看图形结果。输入代码后，按 "Enter" 键，得到的图像结果如图 12.4 所示。



图 12.4 转换后的图像

step 10 计算图像边界，并在边界线上添加直线。在命令窗口中输入下面的程序代码：

```
>> dim = size(BW);
% horizontal beam
col1 = 4;
row1 = min(find(BW(:,col1)));
% angled beam
row2 = 12;
col2 = min(find(BW(row2,:)));
>> boundary1 = bwtraceboundary(BW, [row1, col1], 'N', 8, 70);
% set the search direction to counterclockwise, in order to trace downward.
boundary2 = bwtraceboundary(BW, [row2, col2], 'E', 8, 90,'counter');
imshow(RGB); hold on;
% apply offsets in order to draw in the original image
plot(offsetX+boundary1(:,2),offsetY+boundary1(:,1),'g','LineWidth',2);
plot(offsetX+boundary2(:,2),offsetY+boundary2(:,1),'g','LineWidth',2);
```

step 11 查看图形结果。输入代码后，按"Enter"键，得到的图像结果如图12.5所示。

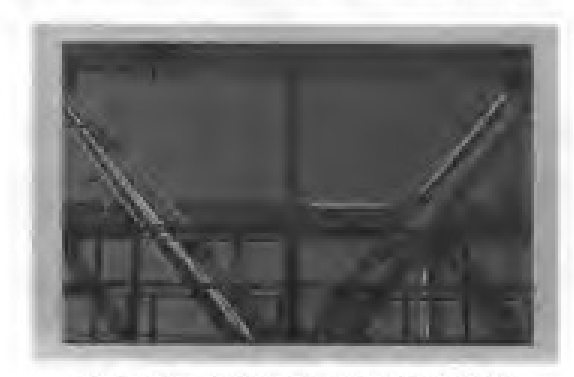


图12.5　向图像文件中添加边界线

step 12 计算两个边界线之间的夹角。在命令窗口中输入下面的程序代码：

```
ab1 = polyfit(boundary1(:,2), boundary1(:,1), 1);
ab2 = polyfit(boundary2(:,2), boundary2(:,1), 1);
>> vect1 = [1 ab1(1)]; % create a vector based on the line equation
vect2 = [1 ab2(1)];
dp = dot(vect1, vect2);
% compute vector lengths
length1 = sqrt(sum(vect1.^2));
length2 = sqrt(sum(vect2.^2));
% obtain the larger angle of intersection in degrees
angle = 180-acos(dp/(length1*length2))*180/pi;
```



由于在前面步骤中获取的边界数据并不完全处在一条直线上，因此，首先使用polyfit命令来进行数据拟合，然后使用线性代数的定义来求取夹角，得出夹角结果。

step 13 计算两个边界线之间的交点。在命令窗口中输入下面的程序代码：

```
intersection = [1 ,-ab1(1); 1, -ab2(1)] \ [ab1(2); ab2(2)];
% apply offsets in order to compute the location in the original,
% i.e. not cropped, image.
intersection = intersection + [offsetY; offsetX]
```

step 14 显示计算结果。在命令窗口中输入下面的程序代码：

```
inter_x = intersection(2);
inter_y = intersection(1);
% draw an "X" at the point of intersection
plot(inter_x,inter_y,'yx','LineWidth',2);
text(inter_x-60, inter_y-30, [sprintf('%1.3f',angle),'{\circ}'],...
     'Color','y','FontSize',14,'FontWeight','bold');
interString = sprintf('(%2.1f,%2.1f)', inter_x, inter_y);
text(inter_x-10, inter_y+20, interString,...
     'Color','y','FontSize',14,'FontWeight','bold');
```

step 15 查看图形结果。输入代码后，按"Enter"键，得到的图像结果如图12.6所示。



图 12.6 显示计算结果

12.6 小结

在本章中，主要向读者介绍了打开外部文件、关闭外部文件、读取二进制文件、写入二进制文件、读取文本文件、写入文本文件和处理图像对象的内容，通过这些基础内容读者可以了解MATLAB中如何进行文件输入和输出。在后面的章节中，将介绍如何使用MATLAB的编译器。

◆ ◆ ◆

第13章 MATLAB编译器

本章包括

- 安装编译器
- 配置编译器
- 编译命令
- 编译过程
- 创建独立的应用程序

前面介绍了MATLAB在各个方面的具体运用。读者也许已经对其强大的功能有所了解，但是也许希望MATLAB能够更快地运行程序代码，或者希望获得可摆脱MATLAB运行环境而独立运行的可执行文件。为了满足用户这方面的需求，MATLAB提供编译器Complier 4.0组件，可以使用该编译器完成上面的任务。在本章中，将向读者详细介绍MATLAB编译器的知识。

13.1 编译器概述

在MATLAB7.0中提供的编译器是Complier 4.0。该编译器将用户在MATLAB中编写的M文件作为输入参数，产生可以重新分配并独立运行的应用程序或者软件组件。通过编译器产生的这些应用程序都是与平台相关的。本节将详细介绍关于该编译器的主要功能和注意事项。

13.1.1 编译器的功能

在MATLAB7.0中，Complier 4.0可以产生下面几种应用程序或者软件组件。

- **独立运行的程序：** 独立运行的程序软件就是在其运行的过程中可以不需要MATLAB7.0软件的同时运行，它们可以在没有安装MATLAB7.0的机器上运行。
- **C和C++共享库（在Microsoft Windows操作系统中为动态连接库DLL）：** 这些共享库可以在没有安装MATLAB7.0的用户机器上运行。
- **Excel附件：** 需要安装MATLAB7.0Bulider for Excel。
- **COM对象：** 需要安装MATLAB7.0Bulider for COM。

MATLAB的编译器支持所有的MATLAB功能和对象，用户不需对私人的函数或者方法进行特殊的处理，Complier 4.0都可以直接处理这些文件。限于篇幅，在本章中将主要介绍前面两种类型的应用程序。



在MATLAB中，有些工具箱将无法和Complier 4.0兼容，对于有些工具箱的特殊函数或者功能，Complier 4.0也无法兼容。如果需要了解工具箱的兼容性，请查找对应的MATLAB Compiler product网站上的内容。

13.1.2 Complier 4.0的性能改进

在MATLAB7.0中，选用的编译器是Complier 4.0。相对于以前的版本，该编译器在性能和功能上

有了很大的改进。在本小节中将主要介绍该编译器在性能上的改进，这样熟悉以前版本的读者可以了解到差异。

- Compiler 4.0 使用MATLAB的新组件 MATLAB Component Runtime (MCR)，而不用MATLAB C/C++ Math and Graphics资源库。MCR是共享资源库中的独立运行组件，该组件可以执行经过编译器编译过的M文件代码。
- 在Compiler 4.0中，只产生接口函数的代码；而在之前的编译器版本中，编译器将产生整个M文件的编译代码。
- Compiler 4.0删除了某些以前编译器版本中产生代码和格式的参数选项，提高了使用编译器的使用便捷性。
- Compiler 4.0并不能加速应用程序，经过编译器编译过的程序代码和在MATLAB环境中运行的应用程序速度基本相当，经过编译的程序代码运行速度和JIT加速器加速的程序代码速度相同。
- 到目前为止，用户只能在Microsoft Windows和 Linux操作系统中使用Compiler 4.0。



关于Complier 4.0的其他性能的说明，感兴趣的读者请查看MATLAB7.0中关于Complier4的帮助文件。

13.2 编译器的安装和配置

在实现MATLAB编译器的各种功能之前，需要首先安装MATLAB的编译器以及其他程序语言的编译器。本节将主要介绍编译器的安装和配置。

13.2.1 前提准备

当用户首次使用程序编译器的时候，MATLAB自动对其进行适当的配置，如果用户对程序编译器有特殊的要求，可以自行手动设置编译器的配置。

step 1 安装下面任何一种可以和MATLAB兼容的ANSI C/C++编译器。

- **LCC**：MATLAB7.0自带的编译器，只能编译C代码，不能编译C++。
- **Borland C++**：可以接受的版本为5.3、5.4、5.5和5.6。
- **Microsoft Visual C/C++**：可以接受的版本为6.0、7.0和7.1。



在默认情况下，只要用户安装了MATLAB7.0，就可以编译C语言程序代码；如果编译某些C++代码，则需要安装另外两个编译器中的一种。根据笔者的经验，为了在后面步骤中方便使用编译器，建议使用完全安装。

step 2 安装MATLAB7.0的Complier 4.0。

在默认情况下，MATLAB7.0中的Complier的安装过程包含在MATLAB7.0的安装过程中，当用户选择的是Typical（典型）安装模式时，MATLAB7.0 Complier会自动被选中。当用户选择的是自定义安装模式的时候，Complier选项也会被默认选中，只需要选中该选项，就可以安装Complier 4.0组件，如图13.1所示。

图 13.1 安装 Complier 4.0

13.2.2 配置编译器

前面做好了安装编译器的准备工作，在本小节中将主要介绍如何对编译器进行配置。由于编译器的配置与用户所使用的系统属性有关，本章仅仅介绍在没有安装其他编译器的情况下，如何正确配置编译器。对于安装了其他类型编译器的读者，其设置工作和本小节类似。

例 13.1 在 MATLAB 中对编译器应用程序 MEX 进行配置。

step 1 启动配置。在 MATLAB 的命令窗口中输入 mex -setup，系统出现下面的提示：

```
Please choose your compiler for building external interface (MEX) files:
Would you like mex to locate installed compilers [y]/n?
```

step 2 启动 MATLAB 的自动定位系统。对于上面的提示内容，如果用户选择 y，mex 将会自动搜索外部编译器的类型、版本以及所在的路径。MATLAB 会给出搜索结果，也就是系统所安装的所有外部编译器，并提示用户输入对应的数字：

```
Select a compiler:
[1] Lcc C version 2.4 in D:\SOFTWARE\MATLAB7.0\sys\lcc
[0] None
Compiler:
```

step 3 确定选择的编译器类型。由于在笔者的系统中，没有安装其他的编译器，因此只有两个选项。当用户选择默认的编译器后，系统会提示用户确定选择：

```
Please verify your choices:
Compiler: Lcc C 2.4
Location: D:\SOFTWARE\MATLAB7.0\sys\lcc
Are these correct?([y]/n):
```

step 4 结束配置。如果上面的定位信息没有错误，可以键入 y，结束编译器的配置工作，MATLAB 会显示结束信息：

```
Try to update options file: C:\Documents and Settings\jackchen\Application
Data\MathWorks\MATLAB\R14\mexopts.bat
From template:                          D:\SOFTWARE\MATLAB7.
0\BIN\WIN32\mexopts\lccopts.bat

Done . . .
```



按照上面步骤完成的配置具有永久性，也就是说，这些配置不会随着用户关闭程序而失效。这些配置工作又是可以修改的，用户可以随时根据版本的情况，重新对编译器进行配置。

例 13.2 验证编译器的配置工作，编译 MEX 文件。

step 1 创建编译文件。前面步骤已经对编译器进行了对应的配置，在本步骤中，需要验证这种配置工作是否正确。选择路径 \MATLAB7.0\extern\examples\mex 中的 yprime.c 文件，将其复制到当前路径中，该文件的具体代码如下：

```
#include <math.h>
#include "mex.h"
/* Input Arguments */
#define  T_IN prhs[0]
#define  Y_IN prhs[1]
/* Output Arguments */
#define  YP_OUT     plhs[0]
#if !defined(MAX)
#define  MAX(A, B)  ((A) > (B) ? (A) : (B))
#endif
#if !defined(MIN)
#define  MIN(A, B)  ((A) < (B) ? (A) : (B))
#endif
#define PI 3.14159265
static   double     mu = 1/82.45;
static   double     mus = 1 - 1/82.45;
static void yprime(
             double  yp[],
             double  *t,
             double  y[]
             )
{
    double    r1,r2;
    r1 = sqrt((y[0]+mu)*(y[0]+mu) + y[2]*y[2]);
    r2 = sqrt((y[0]-mus)*(y[0]-mus) + y[2]*y[2]);
    /* Print warning if dividing by zero. */
    if (r1 == 0.0 || r2 == 0.0 ){
    mexWarnMsgTxt("Division by zero!\n");
    }
    yp[0] = y[1];
     yp[1] = 2*y[3]+y[0]-mus*(y[0]+mu)/(r1*r1*r1)-mu*(y[0]-mus)/
(r2*r2*r2);
    yp[2] = y[3];
    yp[3] = -2*y[1] + y[2] - mus*y[2]/(r1*r1*r1) - mu*y[2]/(r2*r2*r2);
    return;
}
void mexFunction( int nlhs, mxArray *plhs[],
           int nrhs, const mxArray*prhs[] )
    {
    double *yp;
    double *t,*y;
    unsigned int m,n;
    /* Check for proper number of arguments */
    if (nrhs != 2) {
```

```
mexErrMsgTxt("Two input arguments required.");
} else if (nlhs > 1) {
mexErrMsgTxt("Too many output arguments.");
}
/* Check the dimensions of Y.  Y can be 4 X 1 or 1 X 4. */
m = mxGetM(Y_IN);
n = mxGetN(Y_IN);
if (!mxIsDouble(Y_IN) || mxIsComplex(Y_IN) ||
(MAX(m,n) != 4) || (MIN(m,n) != 1)) {
mexErrMsgTxt("YPRIME requires that Y be a 4 x 1 vector.");
}
/* Create a matrix for the return argument */
YP_OUT = mxCreateDoubleMatrix(m, n, mxREAL);
 /* Assign pointers to the various parameters */
yp = mxGetPr(YP_OUT);
t = mxGetPr(T_IN);
y = mxGetPr(Y_IN);
/* Do the actual computations in a subroutine */
yprime(yp,t,y);
return;
}
```

step 2 查看对应的M文件。上面的程序代码是MATLAB7.0自带的MEX文件，对应的文件是yprime.m，其具体代码如下：

```
function yp = yprime(t,y)
mu = 1/82.45;
mus = 1-mu;
r1 = norm([ y(1)+mu, y(3)]);    % Distance to the earth
r2 = norm([ y(1)-mus, y(3)]);   % Distance to the moon
yp(1) = y(2);
yp(2) = 2*y(4) + y(1) - mus*(y(1)+mu)/r1^3 - mu*(y(1)-mus)/r2^3;
yp(3) = y(4);
yp(4) = -2*y(2) + y(3) - mus*y(3)/r1^3 - mu*y(3)/r2^3;
%yp = yp';
```



上面代码的含义并不复杂，熟悉MATLAB编程的读者会很轻松地了解该代码的含义，在这里就不详细介绍该代码的含义了。

step 3 直接运行MEX文件。在MATLAB的命令窗口中输入下面的程序代码：

```
>> mex yprime.c
>> yprime(1,1:4)
```

step 4 查看程序代码的结果。输入代码后，按"Enter"键，得到的结果如下：

```
ans =
    2.0000    8.9685    4.0000   -1.0947
```



在上面的程序代码中，首先使用mex yprime.c命令代码，MATLAB会将yprime.c生成yprime.dll文件，然后通过yprime(1,1:4)调用该yprime.dll文件，并得出结果。

step 5 查看 yprime.dll 文件的定位信息。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> which yprime
D:\SoftWare\MATLAB7.0\work\yprime.dll
```



从上面的结果中可以看出，对于 MEX 文件，该编译器可以正常编译出执行文件，并将该执行文件保存在原来 MEX 文件相同路径中。

例 13.3 验证编译器的配置工作，编译 M 文件。

step 1 创建编译文件。前面步骤中已经使用该编译器编译了 MEX 文件，在本步骤中将需要检验编译器编译 M 文件的功能，其具体代码如下：

```
function mymex
t=1;
y=1:4;
mu = 1/82.45;
mus = 1-mu;
r1 = norm([ y(1)+mu, y(3)]);    % Distance to the earth
r2 = norm([ y(1)-mus, y(3)]);   % Distance to the moon
yp(1) = y(2);
yp(2) = 2*y(4) + y(1) - mus*(y(1)+mu)/r1^3 - mu*(y(1)-mus)/r2^3;
yp(3) = y(4);
yp(4) = -2*y(2) + y(3) - mus*y(3)/r1^3 - mu*y(3)/r2^3;
yp=yp'
```



上面的程序代码是对前面步骤中提到的 yprime 的 M 文件代码的修改，主要减少对应函数中的参数输入情况。

step 2 编译该文件。在 MATLAB 的命令窗口中输入下面的代码：

```
>> mcc -m mymex.m
```

step 3 运行可执行文件。上面的程序代码将会生成 mymex.exe 的可执行文件，可以在 DOS 环境下运行该可执行文件，具体的情况如图 13.2 所示。

图 13.2 在 DOS 条件下执行文件



从上面运行的结果可以看出，在DOS环境中执行exe文件得到的结果和前面步骤中在MATLAB环境中运行的结果相同，表示编译成功。

前面已经介绍配置默认的LCC编译器，如果在用户的系统中安装了其他编译器，则需要使用mbulid命令进行配置。下面以安装了Microsoft Visual C/C++ 6.0的系统为例，简要列出安装的过程，具体的过程如下：

```
mbuild -setup
Please choose your compiler for building standalone MATLAB
applications:
Would you like mbuild to locate installed compilers [y]/n? n
Select a compiler:
[1] Borland C++Builder version 6.0
[2] Borland C++Builder version 5.0
[3] Borland C++Builder version 4.0
[4] Borland C++Builder version 3.0
[5] Borland C/C++ version 5.02
[6] Borland C/C++ version 5.0
[7] Borland C/C++ (free command line tools) version 5.5
[8] Lcc C version 2.4
[9] Microsoft Visual C/C++ version 7.1
[10] Microsoft Visual C/C++ version 7.0
[11] Microsoft Visual C/C++ version 6.0
[0] None
Compiler: 11
Your machine has a Microsoft Visual C/C++ compiler located at
D:\Applications\Microsoft Visual Studio. Do you want to use this
compiler [y]/n? y
Please verify your choices:
Compiler: Microsoft Visual C/C++ 6.0
Location: D:\Applications\Microsoft Visual Studio
Are these correct?([y]/n): y
Try to update options file:
C:\WINNT\Profiles\username\Application
Data\MathWorks\MATLAB\R14\compopts.bat
From template:
\\sys\MATLAB\BIN\WIN32\mbuildopts\msvc60compp.bat
Done . . . .
Updated ...
```



从上面的过程中可以看出，配置外部程序代码编译器和配置默认编译器的过程大致相同，只是涉及的具体选项不同。最后，上面所有的配置结果都是在Windows操作系统中演示的，使用UNIX操作系统的读者请查看对应的帮助文件。

13.3 编译过程

在详细介绍如何在MATLAB中使用编译命令之前，笔者认为有必要大致了解在MATLAB中编译文件的基本原理，这样在编译出现问题的时候，可以根据基本原理来进行修改。本节将简要介绍MATLAB编译过程。

13.3.1 安装 MCR

MCR 是 MATLAB Component Runtime 的缩写，Complier 4.0 编译器采用该技术。其具体含义是一组用来保证 M 文件执行的独立的共享库，MCR 对 MATLAB7.0 语言提供完全的支持。如果用户希望使用 MATLAB 编译器生成的组件，必须安装 MCR，下面介绍其安装的详细过程。

例 13.4 在 Windows 操作系统中安装 MCR 组件。

step 1 复制安装文件。在默认路径\toolbox\compiler\deploy\win32 中找到 MCRInstaller 文件，然后将该文件复制到用户自行设定的路径中，本例中选择的是 E:\Study\Matlab。

step 2 运行安装文件。双击 MCRInstaller.exe 文件，打开安装界面，如图 13.3 所示。



图 13.3 安装 MCR 的界面

step 3 选择安装路径。单击操作界面中的"Next"按钮，然后选择路径，如图 13.4 所示。

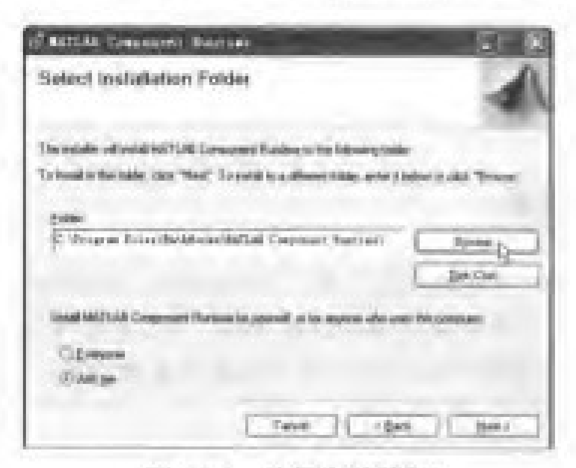


图 13.4 选择安装路径



在上面的对话框中，可以单击"Browse"按钮自行选择安装的路径，也可以单击"Disk cost"按钮，查看安装所需要消耗的磁盘空间。

step 4 运行安装。单击操作界面中的"Next"按钮，系统开始按照用户设定的路径和其他属性进行安装，如图 13.5 所示。



图 13.5　MCR 安装过程界面

当安装结束后，系统提示安装完成，单击对应的按钮就可以完成整个安装过程。如果用户希望修复或者卸载MCR，也可以双击上面的MCRInstaller.exe文件，MATLAB会自动检索该系统中是否安装MCR。如果检测的结果表明在该系统中安装过MCR，系统会显示修复或者卸载MCR，如图13.6所示。



图 13.6　修复或者卸载 MCR 的界面

13.3.2　代码的编译过程

在MATLAB7.0中，Complier 4.0编译器采用了Component Technology File（CTF）的存档方案来组织配置文件包，所有的M文件均采用高级加密标准（AES）进行了密码为1024位的加密，保存为CTF格式。每个由MATLAB编译器产生的应用程序或者共享库均有一个与之对应的CTF存档，包括所有基于MATLAB7.0的M文件或者MEX文件等。

在MATLAB7.0 编译器中，生成独立文件或者程序组件的过程完全是自动的。为了能够生成一个独立运行的MATLAB应用程序，只需要提供一系列用来构成应用程序的M文件程序代码，编译器就会自动运行对应的程序过程。

在本小节中，将以MATLAB中的foo.m和bar.m文件为例，演示在MATLAB中编译器编译这两个文件的过程。

例 13.5　演示 MATLAB 中编译器编译文件的过程。

step 1　绘制编译过程的流程图。为了能够让读者对编译器编译文件的过程有一个直观的印象，首先绘制编译过程的流程图，如图13.7所示。

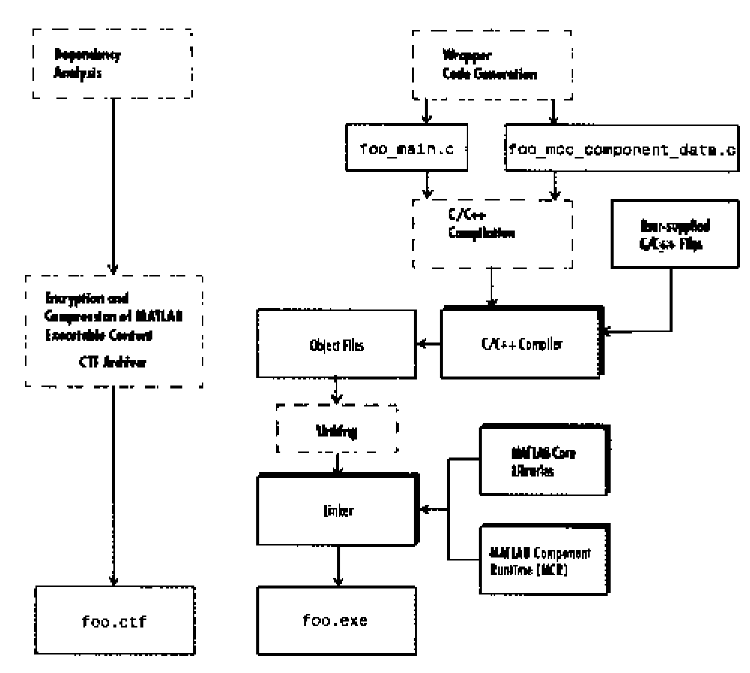


图 13.7 编译文件的流程图

step 2 解读编译过程的流程图。在本步骤中，将结合编译原理来说明编译文件的过程。

- **依赖性分析（Dependency analysis）**：分析判断输入的M文件、MEX文件以及P文件所依赖的函数之间的关系，这些函数包括了输入文件所调用的M文件以及MATLAB提供的函数。
- **代码生成（Code generation）**：生成所有用来产生目标组件的代码，包括从命令行获得的M函数相关的C和C++接口代码。对于共享库和组件来说，这些代码包括所有的接口函数代码。
- **存档生成（Archive creation）**：在依赖性分析中生成的MATLAB7.0可执行文件列表用来生成CTF存档文件，其中包括程序运行时所需组件的数据，存档在加密后被压缩在一个单独的文件中，同时路径信息也被保存。
- **编译（Compilation）**：编译生成的C和C++文件，得到目标代码。对于那些含有用户提供C或者C++代码的目标，这些代码同样也会被编译。
- **链接（Linking）**：将生成的目标文件以及相关的MATLAB的共享库链接起来，生成最终的组件。

编译命令

在MATLAB中提供mcc命令来进行编译工作。本节将主要介绍mcc命令的使用方法和注意事项；关于比较复杂的编译文件、方法以及过程将在后面详细介绍。

编译命令的格式和选项

无论希望生成一种或者多种应用程序，或者希望创建一个C共享库以及软件组件，只要源文件是M文件，都可以使用编译命令mcc来实现，可以在MATLAB环境以及DOS或者UNIX环境中使用该命令。

可以使用一种或者多种MATLAB编译器参数给mcc命令，大部分参数的名字由一个字母组成，用户可以独立地使用两个参数，如下面的示例代码：

```
mcc -m -g myfun
```

除了上面的命令格式之外，MATLAB还提供多种参数来实现不同的编译工作，但是MATLAB中还提供Macros参数来简化编译任务编译参数。它使用户在面对一个简单的问题时可以不再亲自手工组合多种参数来实现某种特定的编译过程，而仅仅使用参数 -m 或者 -l就可以解决问题。例如，对于上面的问题用户可以使用下面的代码：

```
mcc -m myfun
```



对于比较简单的问题，笔者还是建议读者使用具体的编译命令，这样可以方便理解mcc命令的具体参数选项含义。

13.4.2 处理脚本文件

在前面介绍过，在MATLAB中，有脚本文件和函数文件两种M文件类型。在MATLAB中，Compiler 4.0编译器无法编译脚本文件，但是可以编译调用脚本文件的函数文件。因此，不能直接在mcc命令行中引用脚本文件，但是可以在该命令行中引用包含该脚本文件的函数文件。这种方法也是处理脚本文件编译问题的最好方法，因为将脚本文件转换为函数文件是十分简单的，只需要在脚本文件的顶行加入函数声明代码行就可以了。在本小节中，将以一个具体的实例来说明如何处理脚本文件的编译工作。

例 13.6 编译脚本文件 Compli_ball，然后运行编译后的文件。

step 1 查看脚本文件Compli_ball的代码。为了查看后面步骤中编译的结果，在本步骤中，有必要查看该文件的如下具体代码：

```
% Script file Compli_ball.m
% 常数
conv=pi/180;
grav=-9.82;
vo=45;
range=zeros(1,91);
% 计算最长的水平距离
for ii=1:91
     theta=ii-1;
     vxo=vo*cos(theta*conv);
     vyo=vo*sin(theta*conv);
     max_time=-2*vyo/grav;
     range(ii)=vxo*max_time;
end
% 写入水平距离的列表
fprintf('Range versus angle theta:\n');
for ii=1:5:91
    theta=ii-1;
    fprintf('%2d %8.4f\n',theta,range(ii));
end
%计算最长的角度和水平距离
[maxrange index]=max(range);
maxangle=index-1;
fprintf('\n Max range is %8.4f at %2d degrees.\n',maxrange,maxangle);
```



上面的程序代码主要功能是计算抛物线的水平距离，代码内容并不复杂，希望用户自行阅读和理解该程序代码。

step 2 运行脚本文件Compli_ball。在MATLAB的命令窗口中输入"Compli_ball"，查看脚本文件的运行结果：

```
>> Compli_ball
Range versus angle theta:
 0    0.0000
 5   35.8083
10   70.5286
15  103.1059
20  132.5504
25  157.9674
30  178.5847
35  193.7757
40  203.0790
45  206.2118
50  203.0790
55  193.7757
60  178.5847
65  157.9674
70  132.5504
75  103.1059
80   70.5286
85   35.8083
90    0.0000

 Max range is 206.2118 at 45 degrees.
```



之所以需要演示上面的脚本文件，是为了和后面步骤中编译文件得到的结果进行比较，验证编译工作的正确性。

step 3 添加函数声明行。在原来的脚本文件的上端添加下面的程序代码：

```
function Compli_ball
```

添加了上面的程序代码后，保存原来的脚本文件。

step 4 编译脚本文件。在MATLAB的命令窗口中输入下面的程序代码：

```
>> mcc -m Compli_ball.m
```

step 5 运行可执行文件。运行DOS窗口，在上面步骤保存的路径中，运行编译完成的Compli_ball.exe文件，得到的结果如图13.8所示。



从上面的结果可以看出，经过编译得到的结果和在MATLAB中运行的结果是相同的，同时，该例子演示了如何将脚本文件转换为函数文件再编译的方式。

图 13.8　在 DOS 窗口中运行编译文件结果

13.5 创建独立运行的程序

创建独立运行的应用程序是MATLAB编译器的重要功能。在本小节中将使用不同的实例，说明如何在 MATLAB 中编译创建独立运行的程序代码。

13.5.1 编译 M 文件

上面演示过如何将脚本文件编译为可独立执行的程序。在本小节中，将以一个简单的实例说明如何编译函数文件。

例 13.7　编译 M 文件 timings.m，然后运行编译后的文件。

step 1 查看 M 文件 timings.m 的程序代码如下：

```
function timings
%不预先设定数组，使用循环
maxcount=1;
tic;
for jj=1:maxcount
clear square
for ii=1:10000
    square(ii)=ii^2;
end
end
ave1=(toc)/maxcount;
%使用预先设置的数组和循环
maxcount=50;
tic;
for jj=1:maxcount
clear square
square=zeros(1,10000);
for ii=1:10000
    square(ii)=ii^2;
end
```

```
end
ave2=(toc)/maxcount;
%使用向量化结构
maxcount=100;
tic;
for jj=1:maxcount
clear square
ii=1:10000;
    square=ii.^2;
end
ave3=(toc)/maxcount;
%显示计算结果
fprintf('Loop/uninitialized array= %9.5f\n',ave1);
fprintf('Loop/initialized array= %9.5f\n',ave2);-
fprintf('vectorized= %9.5f\n',ave3);
```

step 2 运行 M 文件 timings.m 的代码。在 MATLAB 的命令窗口中输入下面的代码：

```
>> timings
Loop/uninitialized array=   0.34100
Loop/initialized array=   0.00020
vectorized=   0.00010
```



上面的程序代码功能主要是比较向量化程序代码和一般循环语句的程序效率，具体的含义请读者自行查阅关于 M 文件编程的章节。

step 3 编译该 M 文件。在 MATLAB 的命令窗口中输入下面的代码：

```
>> mcc -mv timings.m
```

step 4 查看程序代码的结果。输入代码后，按 "Enter" 键，得到的结果如下：

```
Compiler version: 4.0 (R14)
Parsing file "d:\software\matlab7.0\work\timings.m"
    (Referenced from: "Compiler Command Line").
Parsing file "d:\software\matlab7.0\toolbox\compiler\deploy\matlabrc.m"
    (Referenced from: "Compiler Command Line").
Parsing file "d:\software\matlab7.0\toolbox\compiler\dirname.m"
    (Referenced from: "Compiler Command Line").
Parsing file "d:\software\matlab7.0\toolbox\compiler\deploy\hgrc.m"
    (Referenced from: "d:\software\matlab7.0\toolbox\compiler\deploy
\matlabrc.m").
Parsing file "d:\software\matlab7.0\toolbox\matlab\strfun\str2double.m"
    (Referenced from: "d:\software\matlab7.0\toolbox\compiler\deploy
\matlabrc.m").
Parsing file "d:\software\matlab7.0\toolbox\matlab\general\usejava.m"
    (Referenced from: "d:\software\matlab7.0\toolbox\compiler\deploy
\matlabrc.m").
Parsing file "d:\software\matlab7.0\toolbox\matlab\iofun\fileparts.m"
    (Referenced from: "d:\software\matlab7.0\toolbox\compiler\dirname.m").
Warning: No matching builtin function available for D:\SoftWare\MATLAB7.
0\toolbox\simulink\simulink\set_param.bi
Generating file "timings_main.c".
```

```
Depfun main loop, iteration 1
Processing D:\SoftWare\MATLAB7.0\toolbox\matlab\mcc.enc
1 items added
Processing dependencies...
0 items added
Depfun main loop, iteration 2
Processing dependencies...
0 items added
Processing include files...
2 items added.
Processing exclude list...
0 items removed.
Processing installed directories...
548 items removed.
Generating MATLAB path...
Created 33 path items.
Depfun main loop converged in 2 iterations, total number of files = 8
Generating file "timings_mcc_component_data.c".
Executing command: mbuild -O -v -output 'timings' 'timings_main.c'
'timings_mcc_component_data.c' -link exe
 This is mbuild Copyright 1984-2004 The MathWorks, Inc.
-> Default options filename found in C:\Documents and
Settings\jackchen\Application Data\MathWorks\MATLAB\R14
----------------------------------------------------------------
-> Options file = C:\Documents and Settings\jackchen\Application Data\
MathWorks\MATLAB\R14\compopts.bat
->     COMPILER                 = lcc
->     Compiler flags:
            COMPFLAGS                = -c -Zp8 -I"D:\SoftWare\MATLAB7.0
\sys\lcc\include" -noregistrylookup
          OPTIMFLAGS            = -DNDEBUG
          DEBUGFLAGS            = -g4
          arguments             =
          Name switch           = -Fo
->     Pre-linking commands     =
->     LINKER                   = lcclnk
->     Link directives:
           LINKFLAGS             = -tmpdir "." -L"D:\SoftWare\MATLAB7.0
\sys\lcc\lib" -libpath "D:\SoftWare\MATLAB7.0\extern\lib\win32\lcc"
          LINKFLAGSPOST         = mclmcrrt.lib
          Name directive        = -o "timings.exe"
          File link directive =
          Lib. link directive =
          Rsp file indicator    = @
->     Resource Compiler        =
->     Resource Linker          =
----------------------------------------------------------------
--> "lcc  -c -Zp8 -I"D:\SoftWare\MATLAB7.0\sys\lcc\include" -
noregistrylookup -Fotimings_main.obj -ID:\SoftWare\MATLAB7.
0\extern\include -ID:\SoftWare\MATLAB7.0\simulink\include -DNDEBUG
timings_main.c"
--> "lcc  -c -Zp8 -I"D:\SoftWare\MATLAB7.0\sys\lcc\include" -
noregistrylookup -Fotimings_mcc_component_data.obj -ID:\SoftWare\MATLAB7.
0 \extern\include -ID:\SoftWare\MATLAB7.0\simulink\include -DNDEBUG
timings_mcc_component_data.c"
    Contents of 2541_tmp.rsp:
```

```
  timings_main.obj  timings_mcc_component_data.obj
--> "lcclnk -o "timings.exe" -tmpdir "." -L"D:\SoftWare\MATLAB7.0
\sys\lcc\lib" -libpath "D:\SoftWare\MATLAB7.0\extern\lib\win32\lcc"
@2541_tmp.rsp mclmcrrt.lib"
--> "if exist _lib2541.def del _lib2541.def"
--> "if exist %LIB_NAME_stub.obj del %LIB_NAME_stub.obj"
```

step 5 在 DOS 条件下运行编译后的可执行文件。在上面步骤中保存的路径中，运行编译完成的 timings.exe 文件，得到的结果如图 13.9 所示。



图 13.9　在 DOS 条件下执行可执行文件

13.5.2 编译 M 和 C 的混合文件

除了前面介绍的编译 M 文件之外，MATLAB7.0 中提供的 Complier 4.0 还可以编译 M 文件和 C 文件混合的文件。在本小节中，将以一个简单的例子来演示如何编译 M 和 C 的混合文件。

例 13.8　编译由 M 文件和 C 文件组成的混合文件。

step 1 简述文件组成部分。在本实例中，需要编译的文件如下。

- **mrank.m**：该 M 文件包含函数，该函数返回 1 到 n 维 magic 矩阵的秩。
- **mrankp.c**：在该 C 语言程序代码中，调用 mrank 文件中定义的函数，并返回该函数定义的输出参数数值。

step 2 查看 mrank.m 文件中的程序代码。该 M 文件包含的程序代码如下：

```
function r = mrank(n)
r = zeros(n,1);
for k = 1:n
   r(k) = rank(magic(k));
end
```

step 3 查看 mrankp.c 文件中的程序代码。该 C 文件包含的程序代码如下：

```
#include <stdio.h>
#include <math.h>
#include "libPkg.h"
main( int argc, char **argv )
{
    mxArray *N;          /* Matrix containing n. */
    mxArray *R = NULL;   /* Result matrix. */
    int      n;          /* Integer parameter from command line. */
```

```
    /* 获取输入的参数 */
    if (argc >= 2) {
        n = atoi(argv[1]);
    } else {
        n = 12;
    }
    mclInitializeApplication(NULL,0);
    libPkgInitialize();/* Initialize the library of M-Functions */
    /* 创建矩阵 */
    N = mxCreateScalarDouble(n);
    /* 调用mlfMrank, mrank.m的编译版本 */
    mlfMrank(1, &R, N);
    /* 显示结果 */
    mlfPrintmatrix(R);
    /* 清空矩阵的数值 */
    mxDestroyArray(N);
    mxDestroyArray(R);
    libPkgTerminate(); /* Terminate the library of M-functions */
    mclTerminateApplication();
}
```

step 4 解释 mrankp.c 文件中的程序代码。该 C 文件包含的程序代码含义如下。

- **mxArray *N……else n = 12**：这段程序代码的功能主要是定义 mlfMrank 函数的输入参数。
- **mclInitializeApplication(NULL,0) ……libPkgInitialize()**：该段程序代码的功能是初始化 MCR 参数，并产生 libPkg 共享库。
- **N = mxCreateScalarDouble(n)……mlfMrank(1, &R, N)**：该段程序代码的功能是首先创建包含参数 n 的矩阵，然后调用编译版本的 mrank.m.函数。
- **mlfPrintmatrix(R) ……mclTerminateApplication()**：该段程序代码的功能是输出计算结果，然后释放在程序编译过程中分配的矩阵，最后终止 M 函数的编译。



限于篇幅，在这里只是简要地说明程序代码含义，关于具体程序代码的编写方法，在这里就不详细展开介绍了。

step 5 编译上面的文件。在 MATLAB 的命令窗口中输入下面的程序代码：

```
mcc -W lib:libPkg -T link:exe mrank printmatrix mrankp.
```

step 6 运行可执行文件。在 DOS 环境中运行 mrank.exe 文件，如图 13.10 所示。

step 7 验证编译结果。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> mrank(12)
ans =
     1
     2
     3
     3
     5
     5
     7
     3
```

```
    9
    7
   11
    3
```

图 13.10 在 DOS 环境下运行文件



根据前面步骤中的mrankp.c程序代码，当用户在DOS环境中没有输入任何的参数时，将自动选用参数12，因此在MATLAB命令窗口中输入"mrank(12)"来验证编译结果正确性。

13.5.3 编译包含绘图命令的 M 文件

前面安装了MCR，因此MATLAB可以编译包含绘图命令的M文件。这是因为在安装MCR后，MATLAB会在编译过程中加入图形库。在本小节中，将以一个简单的实例来说明如何编译包含绘图命令的M文件。

例 13.9 编译Areoplot.m文件，该文件的功能主要是绘制在某圆柱体周围的非粘性、无漩涡且不可以压缩的流体的流型线和压力系数图形。

step 1 查看Areoplot.m文件的代码，其具体的程序代码如下：

```
function Areoplot
%定义常数数值
V_i = 25;
G = 10;
a = 1 ;
c =-a*2;
b =a*2;
%迭代次数
n =a*50;
[x,y] =meshgrid([c:(b-c)/n:b],[c:(b-c)/n:b]');
warning off
%Preliminar DATA & purification
for i=1:length(x);
    for k=1:length(x);
        if sqrt(x(i,k).^2+y(i,k).^2)<a;
            x(i,k)=0;
            y(i,k)=0;
        end
```

```
    end
end
% 定义极坐标的数值
rho=sqrt(x.^2+y.^2);
theta=atan2(y,x);
% 创建流线型的函数
z=V_i.*sin(theta).*rho.*(1-(a^2./(rho.^2)))-G*log(rho)/(2*pi);
% 创建图形
n=100;
r=ones(1,n+1)*a;
t=[0:2*pi/n:2*pi];
% 流线型图形 contour(x,y,z,25)
hold on
polar(t,r,'-k')
axis square
title('Stream Lines')
grid off
figure(2)
contour(x,y,z,15)
% 创建环绕在圆柱体的矢量场
x=[-a*2:a/3:a*2];
[x]=meshgrid(x);
y=x';
for i=1:length(x);
    for k=1:length(x);
        if sqrt(x(i,k).^2+y(i,k).^2)<a;
            x(i,k)=0;
            y(i,k)=0;
        end
    end
end
r=sqrt(x.^2+y.^2);
theta=atan2(y,x);
ur=V_i*cos(theta).*(1-a^2./(r.^2));
ut=-V_i*sin(theta).*(1+a^2./(r.^2))+G./(2*pi*r);
u=ur.*cos(theta)-ut.*sin(theta);
v=ur.*sin(theta)+ut.*cos(theta);
% 创建填充后的图形
t_r  = 0:.1:2*pi;
xxx = a*cos(t_r);
yyy = a*sin(t_r);
% 填充后的图形和矢量场图形
figure(2)
hold on
quiver(x,y,u,v)
fill(xxx,yyy,'y')
axis square
title('Speed Vectors')
grid off
warning on
t=0:.1:2*pi;
cp = 1 - 4*sin(t).^2 + 2* G / (pi*a*V_i) *sin(t) - (2* G/ (pi*a*V_i) )^2 ;
cp_sim = 1 - 4*sin(t).^2 ;
L = - 1.225*V_i*G;
L = num2str(L);
```

```
L = strcat('Kutta Joukowski Lift:  ',L,' [N]');
figure(3)
plot(t,cp,t,cp_sim,'--r')
axis([ 0 2*pi min(cp) max(cp_sim)])
title('Pressure coefficient around the surface (standard air density)
')
xlabel('Theta (angle with orizonthal)')
ylabel('C_p')
legend('Lifting solution','Symmetrical solution')
grid on
```



上面的程序代码分别绘制了对应物理参数的图形，具体的代码含义就不在这里解释了，请读者自行分析。

step 2 编译 Areoplot.m 文件。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> mcc -mv Areoplot.m
```

step 3 查看程序代码的结果。输入代码后，按 "Enter" 键，得到的结果如下：

```
Compiler version: 4.0 (R14)
Parsing file "d:\software\matlab7.0\work\areoplot.m"
    (Referenced from: "Compiler Command Line").
Parsing file "d:\software\matlab7.0\toolbox\compiler\deploy\matlabrc.m"
    (Referenced from: "Compiler Command Line").
Parsing file "d:\software\matlab7.0\toolbox\compiler\dirname.m"
    (Referenced from: "Compiler Command Line").
Parsing file "d:\software\matlab7.0\toolbox\matlab\graph2d\axis.m"
    (Referenced from: "d:\software\matlab7.0\work\areoplot.m").
Parsing file "d:\software\matlab7.0\toolbox\matlab\graphics\close.m"
    (Referenced from: "d:\software\matlab7.0\work\areoplot.m").
Parsing file "d:\software\matlab7.0\toolbox\matlab\specgraph\contour.m"
    (Referenced from: "d:\software\matlab7.0\work\areoplot.m").

......................................................//省略了调用函数代码

Parsing file "d:\software\matlab7.0\toolbox\matlab\general\usejava.m"
    (Referenced from: "d:\software\matlab7.0\toolbox\compiler\
deploy\matlabrc.m").
Parsing file "d:\software\matlab7.0\toolbox\matlab\iofun\fileparts.m"
    (Referenced from: "d:\software\matlab7.0\toolbox\compiler\dirname.m").

Generating file "areoplot_main.c".
Depfun main loop, iteration 1
Processing D:\SoftWare\MATLAB7.0\toolbox\matlab\mcc.enc
1 items added
Processing D:\SoftWare\MATLAB7.0\toolbox\database\mcc.enc
1 items added
Processing dependencies...
0 items added
Depfun main loop, iteration 2
Processing dependencies...
0 items added
Processing include files...
```

```
2 items added.
Processing exclude list...
0 items removed.
Processing installed directories...
1575 items removed.
Generating MATLAB path...
Created 34 path items.
Depfun main loop converged in 2 iterations, total number of files = 65
Generating file "areoplot_mcc_component_data.c".
Executing command: mbuild -O -v -output 'Areoplot' 'areoplot_main.c'
'areoplot_mcc_component_data.c' -link exe
 This is mbuild Copyright 1984-2004 The MathWorks, Inc.
-> Default options filename found in C:\Documents and
Settings\jackchen\Application Data\MathWorks\MATLAB\R14
----------------------------------------------------------------
->       Options file                    = C:\Documents and
Settings\jackchen\Application Data\MathWorks\MATLAB\R14\compopts.bat
->    COMPILER                   = lcc
->    Compiler flags:
        COMPFLAGS    = -c -Zp8 -I"D:\SoftWare\MATLAB7.0\sys\lcc\include"
-noregistrylookup
         OPTIMFLAGS          = -DNDEBUG
         DEBUGFLAGS          = -g4
         arguments           =
         Name switch         = -Fo
->    Pre-linking commands       =
->    LINKER                     = lcclnk
->    Link directives:
         LINKFLAGS             = -tmpdir "." -L"D:\SoftWare\MATLAB7.0
\sys\lcc\lib" -libpath "D:\SoftWare\MATLAB7.0\extern\lib\win32\lcc"
         LINKFLAGSPOST       = mclmcrrt.lib
         Name directive      = -o "Areoplot.exe"
         File link directive =
         Lib. link directive =
         Rsp file indicator  = @
->    Resource Compiler          =
->    Resource Linker            =
----------------------------------------------------------------
--> "lcc  -c -Zp8 -I"D:\SoftWare\MATLAB7.0\sys\lcc\include" -
noregistrylookup -Foareoplot_main.obj -ID:\SoftWare\MATLAB7.0
\extern\include -ID:\SoftWare\MATLAB7.0\simulink\include -DNDEBUG
areoplot_main.c"
--> "lcc  -c -Zp8 -I"D:\SoftWare\MATLAB7.0\sys\lcc\include" -
noregistrylookup -Foareoplot_mcc_component_data.obj -ID:
\SoftWare\MATLAB7.0\extern\include -ID:\SoftWare\MATLAB7.0
\simulink\include -DNDEBUG areoplot_mcc_component_data.c"
    Contents of 5152_tmp.rsp:
  areoplot_main.obj  areoplot_mcc_component_data.obj
--> "lcclnk -o "Areoplot.exe" -tmpdir "." -L"D:\SoftWare\MATLAB7.0
\sys\lcc\lib" -libpath "D:\SoftWare\MATLAB7.0\extern\lib\win32\lcc"
@5152_tmp.rsp mclmcrrt.lib"
--> "if exist _lib5152.def del _lib5152.def"
--> "if exist %LIB_NAME_stub.obj del %LIB_NAME_stub.obj"
```



之所以在上面的步骤中列出详细的编译过程，是为了让用户体验编译包含图形代码的M文件和一般M文件有什么区别。

step 4 运行编译后的Areoplot.exe文件。在DOS窗口条件下运行编译后的可执行文件，具体情况如图13.11所示。



图 13.11 在 DOS 环境中运行可执行文件



由于在实例中的所有结果都是图形化结果，没有任何的数据结果，因此在DOS环境下不会显示任何结果，当用户查看了所有图形结果并关闭窗口后，DOS窗口会提示用户可以输入下一条命令。

执行编译后的文件后，MATLAB得到的流线型图形如图13.12所示。

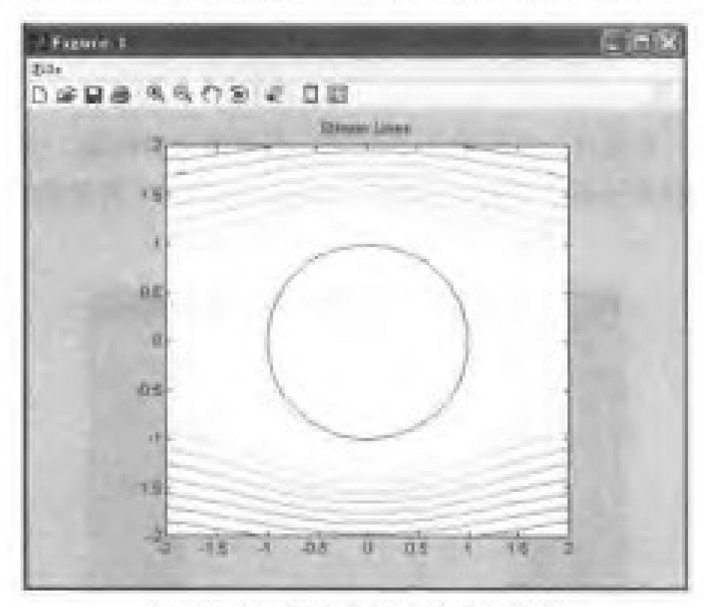


图 13.12 程序绘制的流线型图形

程序绘制的速度向量图形如图13.13所示。

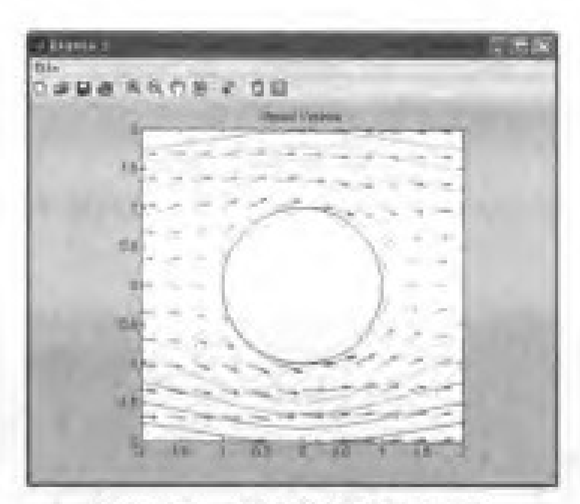


图 13.13　绘制的速度向量图形

程序绘制所得到的表面压力系数图形如图 13.14 所示。

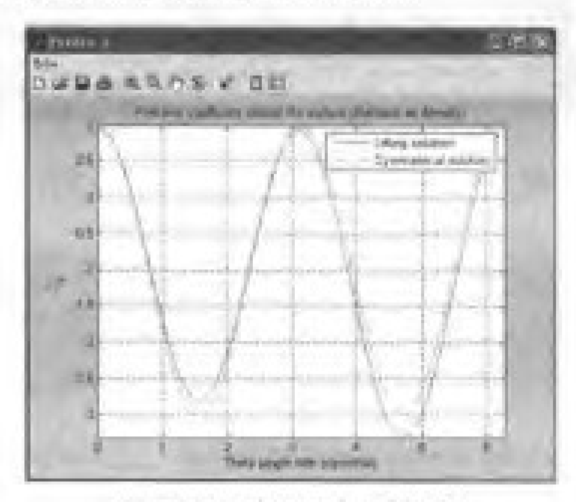


图 13.14　表面压力系数图形

step 5 对比图形界面结果。

需要提醒用户的是，尽管在 DOS 环境下运行可以得到类似的结果，但是得到的图形窗口和在 MATLAB运行程序得到的结果还是有不同的，例如，在MATLAB环境下得到的速度向量图形如图13.15 所示。

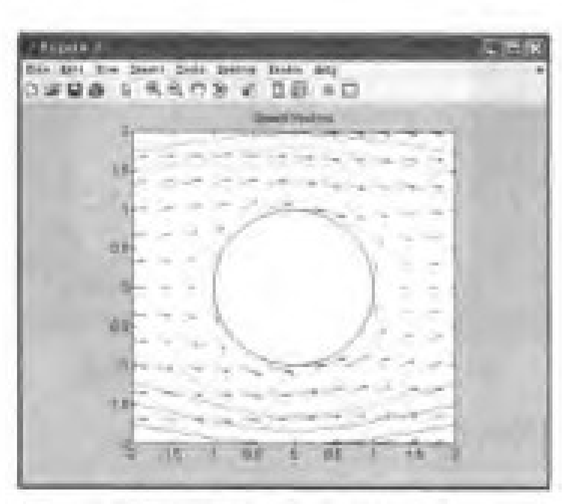
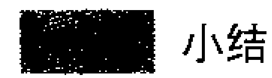


图 13.15　在 MATLAB 环境中得到的图形

 小结

本章主要介绍了如何在MATLAB中安装和配置编译器，介绍了MATLAB中的编译命令和过程，最后还介绍了如何使用编译器来创建独立的应用程序，通过这些内容希望读者能够对MATLAB的编译器有大概的了解。下一章将介绍如何在MATLAB中编写应用程序接口。

◆ ◆ ◆

第14章 应用程序接口

本章包括

- 使用C语言创建MEX文件
- 使用C语言创建MAT文件
- 使用FORTRAN语言创建MAT文件
- MATLAB的引擎技术
- Java接口

前面主要介绍了MATLAB自身的功能和使用方法，但是作为一个优秀工程软件，MATLAB除自身功能强大、环境友好之外，还有很好的开放性。这种开放性体现在MATLAB可以和外部应用程序实现“无缝”结合，提示了专业应用程序接口API。

在本章中主要从下面几个方面来介绍MATLAB应用程序接口。首先介绍MEX文件——外部程序调用接口，在MATLAB中使用C语言或者FORTRAN语言编写的程序代码，用来提供程序运行的效率；然后介绍MAT文件应用程序——数据输入输出接口，向MATLAB输入或者输出数据的程序代码；最后，将介绍MATLAB计算引擎函数库——在MATLAB和其他应用程序中建立客户机/服务器关系，将MATLAB作为计算引擎，在其他应用程序中调用，从而降低应用程序的计算量。在Windows操作系统中，MATLAB支持该系统提供的COM标准，同时支持Java语言，因此MATLAB几乎可以和任何一种软件进行交互。

C语言MEX文件

MEX文件是一种可以在MATLAB中调用的C语言或者FORTRAN语言衍生程序代码，而MEX本身就是MATLAB和Executable两个单词的缩写。通过C语言编写的MEX文件程序代码，经过适当的编译后，生成的目标文件能够被M语言解释器调用执行，在Windows操作系统下这些文件使用后缀dll(Dynamic link library)。MEX文件的使用极为方便，其调用方式与MATLAB的内建函数完全相同，只需要在命令窗口输入对应的文件名称即可。



MEX文件的数据

和其他语言程序代码的编写一样，在MATLAB中用户希望编写MEX程序代码，则首先有必要了解MEX文件中的数据类型，以及这些数据类型和MATLAB对应的数据之间的联系等。在本小节中，将简要介绍MEX中的主要数据类型。

由于在MATLAB中所有的数据都是以矩阵或者阵列（Array）存储的，因此如果使用C语言来编写MEX文件，就必须能够处理对应的数据类型。在C语言中，Array数据类型用mxArray来定义，这种结构体包含的信息有变量类型、维数和数据等。对于数值类型的变量，该结构体说明变最是实数还是复数；对于稀疏类型的变量，该结构体说明变量的下标和最大非零元素；对于构架类型的变量，该结构体说明变量的域名和对应的数值。

根据上面的介绍，该结构体的存储信息如下。

- **复数双精度数值矩阵：**该结构体将存储这些变量的双精度类属性、维数，双精度变量的实部变量和虚部变量，实部变量指针pr，虚部变量指针pi。

- **其他数值矩阵：** 带符号和不带符号的8位、16位和32位整数矩阵，单精度浮点数矩阵存储方式和双精度矩阵相同。
- **字符串：** 字符串类，每个字符串用16位ASCII Unicode码表示。
- **元胞数组变量：** 每一个mxArray结构体对应一个元胞变量，这样就可以保证每一个元胞变量能够存放任何一个数据类型的MATLAB变量；存储方式和数值数组相同，但是部分数组数值包含了mxArray结构体的指针信息。
- **构架数据类型：** 单构架的存储方式和数值矩阵的方法相同，每个数值域对应着依此为名称存储的mxArray结构体。

14.1.2 MEX 文件的结构

尽管在前面介绍过，在MATLAB中调用MEX文件会很简单，但是使用一般格式编写的C语言程序代码并不能直接编译成可以被MATLAB调用的MEX文件，只有符合某种特殊格式的C程序代码才能编译成为MEX文件。在本小节中，将以一个简单的例子来说明MEX的典型结构。

例 14.1 编写C语言MEX程序代码，实现的功能是将输入变量乘以2然后输出。

step 1 编写对应的C语言程序代码。在任何可以编译C语言程序代码的编译器中，输入下面的程序代码：

```
#include <math.h>
void timestwo(double y[], double x[])
{
  y[0] = 2.0*x[0];
  return;
}
```



上面的程序代码十分简单，在这里就不详细分析了。安装了C语言编译器的读者，可以在自己的机器上自行编译和运行上面的程序代码。

step 2 编写对应的C语言MEX程序代码。在C语言编译器里编写对应的C语言MEX程序代码，其对应的代码如下：

```
#include "mex.h"
void timestwo(double y[], double x[])
{
  y[0] = 2.0*x[0];
}
void mexFunction(int nlhs, mxArray *plhs[], int nrhs,
                 const mxArray *prhs[])
{
  double *x, *y;
  int mrows, ncols;

  /* 检测输入参数的个数 */
  if (nrhs != 1) {
    mexErrMsgTxt("One input required.");
  } else if (nlhs > 1) {
    mexErrMsgTxt("Too many output arguments");
  }
  /* 确保输入参数是标量，并且是正值 */
```

```
  mrows = mxGetM(prhs[0]);
  ncols = mxGetN(prhs[0]);
  if (!mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
      !(mrows == 1 && ncols == 1)) {
    mexErrMsgTxt("Input must be a noncomplex scalar double.");
  }
  /* 创建返回参数的属性 */
  plhs[0] = mxCreateDoubleMatrix(mrows,ncols, mxREAL);
  /* 为输入参数和输出参数分配指针 */
  x = mxGetPr(prhs[0]);
  y = mxGetPr(plhs[0]);
  /* 调用子函数 */
  timestwo(y,x);
}
```



当输入上面的程序代码后，将文件保存为twotimes.c。但是不要在C语言编译器中编译上面的文件代码，该文件需要在MATLAB环境中编译。

step 3 在MATLAB环境中编译并链接上面的文件。在MATLAB的命令窗口中输入下面的程序代码：

```
>> mex timestwo.c
>> which timestwo.dll
```

在上面的程序代码中，首先编译了上面的程序代码，然后定位编译后的文件，得到的编译后的文件信息如下：

```
D:\SoftWare\MATLAB7.0\work\timestwo.dll
```

step 4 演示程序代码功能。在MATLAB的命令窗口中输入下面的程序代码：

```
>> x = 2;
y = timestwo(x)
y =
     4
```



从上面的结果可以看出，编译后的程序代码得到的结果和原来的C语言程序代码功能完全相同，表示MATLAB的编译成功。

下面将以上面简单的例子来说明MEX的一般结构。

- **#include "mex.h"……y[0] = 2.0*x[0]**：这段程序代码的主要功能是首先进行头文件声明，声明与MEX交互所需要的宏、函数等，然后列出计算子程序代码，这段程序代码和原始的C语言程序代码几乎没有差别。
- **void mexFunction(int nlhs, mxArray *plhs[], int nrhs const mxArray *prhs[])……mexErrMsgTxt("Too many output arguments")**：该段程序代码的功能是首先声明接口子程序代码，然后检测程序代码输入参数的个数。
- **mrows = mxGetM(prhs[0]) ……mexErrMsgTxt("Input must be a noncomplex scalar double.")**：该段程序代码的主要功能是检测输入参数是否是标量，如果不是标量，则显示提示错误信息，并退出程序代码。

- plhs[0] = mxCreateDoubleMatrix(mrows,ncols, mxREAL)：该程序代码的功能是为返回参数创建存储的数组空间。
- x = mxGetPr(prhs[0]) …… y = mxGetPr(plhs[0])：该段程序代码的功能是为输入和输出参数分配指针变量。
- timestwo(y,x)：该段程序代码的功能是调用上面步骤中的计算子程序代码。



上面的分析过程中，大部分的程序代码是使用C语言创建MEX文件所必需的，但是也有部分程序代码是这个实例所特有的，希望读者仔细区分。

根据上面的分析，读者可以了解到C语言MEX文件结构的如下信息。

- 首先，C语言MEX程序代码文件必须以#include "mex.h"开始，确定程序数据接入和交互被正确声明。
- C语言MEX程序代码文件由计算子例程（Computational routine）和接口子程序（Gateway routine）两个相互独立的子程序组成。其中，计算子例程的功能是完成所需的计算，它和具有相同功能的一般C源程序文件几乎相同。接口子程序的功能则是计算子程序和MATLAB的接口，用户实现两个不同内存空间中的通信。
- 接口子程序的名称只能是带有规范参数的mexFunction。



尽管通过本小节的介绍，可以看出创建MEX文件的方法并不复杂，但是由于MATLAB本身是高效的系统，如果不是出于实际需要，不建议使用MEX文件的方法。

在本小节的最后，提供另外一组程序代码实现上面程序的功能。该程序代码中使用一种特殊的API函数mxGetScalar，该函数将直接返回数组中数值，而不需要使用数值副本的指针变量，具体的程序代码如下：

```
#include "mex.h"
void timestwo_alt(double *y, double x)
{
  *y = 2.0*x;
}
void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[])
{
  double *y;
  double x;
  /* Create a 1-by-1 matrix for the return argument. */
  plhs[0] = mxCreateDoubleMatrix(1, 1, mxREAL);
  /* Get the scalar value of the input x. */
  /* Note: mxGetScalar returns a value, not a pointer. */
  x = mxGetScalar(prhs[0]);
  /* Assign a pointer to the output. */
  y = mxGetPr(plhs[0]);

  /* Call the timestwo_alt subroutine. */
  timestwo_alt(y,x);
}
```



为了节省篇幅，在上面的程序代码中省略了处理输入或者输出参数的程序代码段，这个段落和原来的程序代码相同。

14.1.3 MEX 文件的实例

前面介绍了 MEX 文件的数据类型和程序结构，在本小节中将分别介绍几个不同的实例，加深对 MEX 文件的理解。

例 14.2 编写 C 语言的 MEX 文件，向 MEX 文件中传递构架和元胞数据。

step 1 打开在系统中安装的任意一个 C 语言开发工具，在该开发工具中输入下面的程序代码。

```
/*
 * ==============================================================
 * phonebook.c
 * Takes a (MxN) structure matrix and returns a new structure
 * (1x1) containing corresponding fields:for string input, it
 * will be (MxN) cell array; and for numeric (noncomplex, scalar)
 * input, it will be (MxN) vector of numbers with the same
 * classID as input, such as int, double etc..
 *
 * ==============================================================
#include "mex.h"
#include "string.h"
#define MAXCHARS 80   /* max length of string contained in each
                         field */
/* The gateway 子程序 */
void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[])
{
  const char **fnames;        /* 域名的指针变量 */
  const int  *dims;
  mxArray    *tmp, *fout;
  char       *pdata;
  int        ifield, jstruct, *classIDflags;
  int        NStructElems, nfields, ndim;

  /* 确保正确的输入和输出变量 */
  if (nrhs != 1)
    mexErrMsgTxt("One input required.");
  else if (nlhs > 1)
    mexErrMsgTxt("Too many output arguments.");
  else if (!mxIsStruct(prhs[0]))
    mexErrMsgTxt("Input must be a structure.");
  /* 获取输入参数数值 */
  nfields = mxGetNumberOfFields(prhs[0]);
  NStructElems = mxGetNumberOfElements(prhs[0]);
  /* 为保存变量 classIDflags 分配内存 */
  classIDflags = mxCalloc(nfields, sizeof(int));
  /* 检测正确的数据类型、数据类型的兼容性，并返回每个属性域的域名 */
  for (ifield = 0; ifield < nfields; ifield++) {
    for (jstruct = 0; jstruct < NStructElems; jstruct++) {
      tmp = mxGetFieldByNumber(prhs[0], jstruct, ifield);
      if (tmp == NULL) {
```

```
        mexPrintf("%s%d\t%s%d\n",
            "FIELD:", ifield+1, "STRUCT INDEX :", jstruct+1);
        mexErrMsgTxt("Above field is empty!");
      }
      if (jstruct == 0) {
        if ((!mxIsChar(tmp) && !mxIsNumeric(tmp)) ||
            mxIsSparse(tmp)) {
          mexPrintf("%s%d\t%s%d\n",
              "FIELD:", ifield+1, "STRUCT INDEX :", jstruct+1);
          mexErrMsgTxt("Above field must have either "
              "string or numeric non-sparse data.");
        }
        classIDflags[ifield] = mxGetClassID(tmp);
      } else {
        if (mxGetClassID(tmp) != classIDflags[ifield]) {
          mexPrintf("%s%d\t%s%d\n",
              "FIELD:", ifield+1, "STRUCT INDEX :", jstruct+1);
          mexErrMsgTxt("Inconsistent data type in above field!");
        }
        else if (!mxIsChar(tmp) && ((mxIsComplex(tmp) ||
            mxGetNumberOfElements(tmp) != 1))) {
          mexPrintf("%s%d\t%s%d\n",
              "FIELD:", ifield+1, "STRUCT INDEX :", jstruct+1);
          mexErrMsgTxt("Numeric data in above field "
              "must be scalar and noncomplex!");
        }
      }
    }
  }
  /* 为保存指针变量分配内存空间 */
  fnames = mxCalloc(nfields, sizeof(*fnames));
  /* Get field name pointers */
  for (ifield = 0; ifield < nfields; ifield++) {
    fnames[ifield] = mxGetFieldNameByNumber(prhs[0],ifield);
  }
  /* 为输出变量创建结构体数组 */
  plhs[0] = mxCreateStructMatrix(1, 1, nfields, fnames);
  mxFree(fnames);
  ndim = mxGetNumberOfDimensions(prhs[0]);
  dims = mxGetDimensions(prhs[0]);
  for (ifield = 0; ifield < nfields; ifield++) {
    /* Create cell/numeric array */
    if (classIDflags[ifield] == mxCHAR_CLASS) {
      fout = mxCreateCellArray(ndim, dims);
    } else {
      fout = mxCreateNumericArray(ndim, dims,
          classIDflags[ifield], mxREAL);
      pdata = mxGetData(fout);
    }
    /* 从输入中复制数据 */
    for (jstruct = 0; jstruct < NStructElems; jstruct++) {
      tmp = mxGetFieldByNumber(prhs[0],jstruct,ifield);
      if (mxIsChar(tmp)) {
        mxSetCell(fout, jstruct, mxDuplicateArray(tmp));
      } else {
        size_t     sizebuf;
```

```
            sizebuf = mxGetElementSize(tmp);
            memcpy(pdata, mxGetData(tmp), sizebuf);
            pdata += sizebuf;
          }
        }
        /* 设置输出结构的名称 */
        mxSetFieldByNumber(plhs[0], 0, ifield, fout);
      }
    mxFree(classIDflags);
    return;
}
```

在输入上面的程序代码后，将程序代码保存为 phonebook.c，并将该 C 语言程序代码文件复制到 MATLAB 的用户工作路径中。

step 2 编译上面的程序代码。返回 MATLAB 的命令窗口中，在命令窗口中输入下面的程序代码：

```
>>mex phonebook.c
>> dir phonebook.*
```

step 3 查看程序代码的结果。输入代码后，按"Enter"键，得到的结果如下：

```
phonebook.c    phonebook.dll
```

上面的结果表明，MATLAB 已经将上面的程序代码编译为 dll 文件。

step 4 运行程序代码。在 MATLAB 的命令窗口中输入下面的程序代码：

```
friends(1).name = 'Jordan Robert';
friends(1).phone = 3386;
friends(2).name = 'Mary Smith';
friends(2).phone = 3912;
friends(3).name = 'Stacy Flora';
friends(3).phone = 3238;
friends(4).name = 'Harry Alpert';
friends(4).phone = 3077;
phonebook(friends)
```

step 5 查看程序代码的结果。输入代码后，按"Enter"键，得到如下的结果：

```
ans =

    name: {'Jordan Robert'  'Mary Smith'  'Stacy Flora'  'Harry Alpert'}
    phone: [3386 3912 3238 3077]
```



从上面的结果可以看出，在 MATLAB 中创建单构架变量后，使用 phonebook 程序可以显示该构架变量的具体信息。在上面的程序代码中，使用 mxGetField（针对构架变量）和 mxGetCell（针对元胞变量）分别返回 mxArray。数据类型的指针，这是在 MEX 文件中处理构架变量和元胞变量的常见方法。

例 14.3 编写 C 语言的 MEX 文件，在其中调用 MATLAB 的内置函数。

step 1 打开用户在系统中安装的任意一个 C 语言开发工具，在该开发工具中输入下面的程序代码：

```
/*
 * ============================================================
 * sincall.c
 *
 *
 * ============================================================
#include "mex.h"
#define MAX 1000
/* Subroutine for filling up data */
void fill(double *pr, int *pm, int *pn, int max)
{
  int i;
  /* You can fill up to max elements, so (*pr) <= max. */
  *pm = max/2;
  *pn = 1;
  for (i = 0; i < (*pm); i++)
    pr[i] = i * (4*3.14159/max);
}
/* gateway子程序 */
void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[])
{
  int     m, n, max = MAX;
  mxArray *rhs[1], *lhs[1];
  rhs[0] = mxCreateDoubleMatrix(max, 1, mxREAL);
  /* 传递指针变量数组，并填充数组数值 */
  fill(mxGetPr(rhs[0]), &m, &n, MAX);
  mxSetM(rhs[0], m);
  mxSetN(rhs[0], n);
  /* 获取sine曲线数据，并绘制该曲线 */
  mexCallMATLAB(1, lhs, 1, rhs, "sin");
  mexCallMATLAB(0, NULL, 1, lhs, "plot");
  /* 清除分配的内存空间 */
  mxDestroyArray(rhs[0]);
  mxDestroyArray(lhs[0]);
return;
}
```

输入上面的程序代码后，将上面的程序代码保存为sincall.c，然后将该程序代码文件保存到MATLAB的目录路径中。

step 2 编译并运行程序代码。在MATLAB的命令窗口中输入下面的程序代码：

```
>> mex sincall.c
>> sincall
```

step 3 查看图形结果。输入代码后，按"Enter"键，得到的图形如图14.1所示。

在C语言编写的MEX程序代码中，用户可以使用API函数mexCallMATLAB来调用MATLAB中的函数、运算符、M文件或者其他MEX文件。在上面的实例中，首先定义了变量mxArray，同时通过不同的指针变量来传递数据，最后使用mexCallMATLAB来调用sine和plot函数来计算和绘制图形结果。

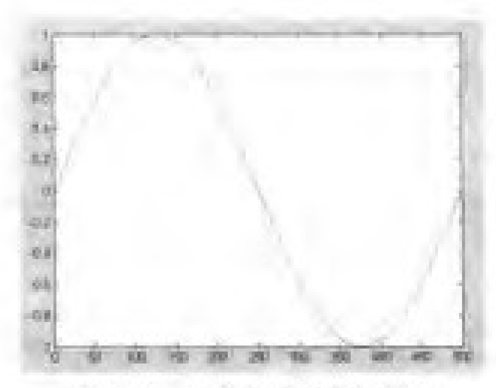


图 14.1 调用程序得到的结果



在MATLAB中，如果用户安装了FORTRAN安装环境，例如Compaq Visual Fortran等，则可以使用FORTRAN语言编写对应的MEX文件。限于篇幅，在本章中就不详细介绍了，请读者自行尝试。

14.2 MAT 文件

MAT文件是用户实现MATLAB和其他应用程序进行数据交换的重要方式和手段。在MATLAB中，MAT文件是使用C语言或者FORTRAN语言编写的专门文件，它可以是MEX文件，也可以是独立的可执行程序。为了便于读写MAT文件，MATLAB提供相应的接口函数MAT函数，MAT文件应用程序就是利用这些mat函数来完成MAT数据文件的读写工作。本节将介绍如何使用C语言和FORTRAN语言编写MEX文件。

14.2.1 使用C语言创建MAT文件

在本小节中，将使用简单的实例来介绍如何使用C语言来创建MAT文件，希望读者从中能够了解MAT应用程序的基本结构和应用的过程。

例 14.4 使用C语言编写创建MAT文件的程序代码。

step 1 打开用户在系统中安装的任意一个C语言开发工具，然后在该开发工具中输入下面的程序代码：

```
/*
 * MAT-file creation program
 *
 *
 * This program demonstrates the use of the following functions:
 *
 *  matClose
 *  matGetVariable
 *  matOpen
 *  matPutVariable
 *  matPutVariableAsGlobal
 *
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```
#include "mat.h"
#define BUFSIZE 256
int main() {
%为MAT文件定义指针
  MATFile *pmat;
%定义结构体的指针
  mxArray *pa1, *pa2, *pa3;
%定义双精度变量
  double data[ 9]  = { 1.0, 4.0, 7.0, 2.0, 5.0, 8.0, 3.0, 6.0, 9.0 };
  const char *file = "mattest.mat";
%定义字符串变量
  char str[ BUFSIZE] ;
  int status;
%以标准的C格式输出MAT文件名称
  printf("Creating file %s...\n\n", file);
%以“写”模式打开名称为file的MAT文件
  pmat = matOpen(file, "w");
  if (pmat == NULL) {
    printf("Error creating file %s\n", file);
    printf("(Do you have write permission in this directory?)\n");
    return(EXIT_FAILURE);
  }
  pa1 = mxCreateDoubleMatrix(3,3,mxREAL);
  if (pa1 == NULL) {
      printf("%s : Out of memory on line %d\n", __FILE__, __LINE__);
      printf("Unable to create mxArray.\n");
      return(EXIT_FAILURE);
  }
  pa2 = mxCreateDoubleMatrix(3,3,mxREAL);
  if (pa2 == NULL) {
      printf("%s : Out of memory on line %d\n", __FILE__, __LINE__);
      printf("Unable to create mxArray.\n");
      return(EXIT_FAILURE);
  }
%将data缓冲区中的内容复制到pa2所指实部的目标缓冲区中
  memcpy((void *)(mxGetPr(pa2)), (void *)data, sizeof(data));
%为pa3所创建字符串的指针
  pa3 = mxCreateString("MATLAB: the language of technical computing");
  if (pa3 == NULL) {
      printf("%s :  Out of memory on line %d\n", __FILE__, __LINE__);
      printf("Unable to create string mxArray.\n");
      return(EXIT_FAILURE);
  }
  status = matPutVariable(pmat, "LocalDouble", pa1);
  if (status != 0) {
      printf("%s :  Error using matPutVariable on line %d\n", __FILE__,
__LINE__);
      return(EXIT_FAILURE);
  }
  status = matPutVariableAsGlobal(pmat, "GlobalDouble", pa2);
  if (status != 0) {
      printf("Error using matPutVariableAsGlobal\n");
      return(EXIT_FAILURE);
  }
  status = matPutVariable(pmat, "LocalString", pa3);
  if (status != 0) {
```

```
        printf("%s :  Error using matPutVariable on line %d\n", __FILE__,
__LINE__);
        return(EXIT_FAILURE);
    }
    /*

    memcpy((void *)(mxGetPr(pa1)), (void *)data, sizeof(data));
    status = matPutVariable(pmat, "LocalDouble", pa1);
    if (status != 0) {
        printf("%s :  Error using matPutVariable on line %d\n", __FILE__,
__LINE__);
        return(EXIT_FAILURE);
    }
    /* 释放所有的内存空间*/
    mxDestroyArray(pa1);
    mxDestroyArray(pa2);
    mxDestroyArray(pa3);
    if (matClose(pmat) != 0) {
      printf("Error closing file %s\n",file);
      return(EXIT_FAILURE);
    }
    /*
     * 再次打开MAT文件，对写入的内容进行验证
     */
    pmat = matOpen(file, "r");
    if (pmat == NULL) {
      printf("Error reopening file %s\n", file);
      return(EXIT_FAILURE);
    }
    /*
     * 读入之前定义的所有数据行
     */
    pa1 = matGetVariable(pmat, "LocalDouble");
    if (pa1 == NULL) {
      printf("Error reading existing matrix LocalDouble\n");
      return(EXIT_FAILURE);
    }
    if (mxGetNumberOfDimensions(pa1) != 2) {
     printf("Error saving matrix: result does not have two dimensions\n");
      return(EXIT_FAILURE);
    }
    pa2 = matGetVariable(pmat, "GlobalDouble");
    if (pa2 == NULL) {
      printf("Error reading existing matrix GlobalDouble\n");
      return(EXIT_FAILURE);
    }
    if (!(mxIsFromGlobalWS(pa2))) {
      printf("Error saving global matrix: result is not global\n");
      return(EXIT_FAILURE);
    }
    pa3 = matGetVariable(pmat, "LocalString");
    if (pa3 == NULL) {
      printf("Error reading existing matrix LocalString\n");
      return(EXIT_FAILURE);
    }
    status = mxGetString(pa3, str, sizeof(str));
```

```
  if(status != 0) {
      printf("Not enough space. String is truncated.");
      return(EXIT_FAILURE);
  }
  if (strcmp(str, "MATLAB: the language of technical computing")) {
    printf("Error saving string: result has incorrect contents\n");
    return(EXIT_FAILURE);
  }
  /* 释放所有的内存空间 */
  mxDestroyArray(pa1);
  mxDestroyArray(pa2);
  mxDestroyArray(pa3);
  if (matClose(pmat) != 0) {
    printf("Error closing file %s\n",file);
    return(EXIT_FAILURE);
  }
  printf("Done\n");
  return(EXIT_SUCCESS);
}
```

在完成上面的程序代码后，将上面的程序代码保存为 matcreat.c 文件。

step 2 编译上面的程序代码。在 MATLAB 的命令窗口中输入下面的程序代码：

```
mex -f $MATLAB\bin\win32\mexopts\msvc60engmatopts.bat matcreat.c
```



上面的编译代码中，msvc60engmatopts.bat 代表的是编译器的类型，在本实例中使用的C语言编译器是Microsoft Visual C++ 6.0,因此需要选择的选项是msvc60engmatopts.bat，可以根据自己的编译器情况来选择对应的编译器类型。

step 3 运行上面的程序代码。上面的程序代码可以在用户的目录路径中创建 mattest.mat 文件，然后在对应的目录路径中双击该文件，在 MATLAB 的命令窗口中输入下面的程序代码：

```
whos -file mattest.mat
  Name                  Size            Bytes  Class
  GlobalDouble          3x3                72  double array (global)
  LocalDouble           3x3                72  double array
  LocalString           1x43               86  char array

Grand total is 61 elements using 230 bytes
```

step 4 查看变量的结果。在 MATLAB 的命令窗口中输入变量名称，查看具体的内容，得到的结果如下：

```
GlobalDouble =
     1     2     3
     4     5     6
     7     8     9
LocalDouble =
     1     2     3
     4     5     6
     7     8     9
```

```
LocalString =
MATLAB: the language of technical computing
```



除了可以编写创建MEX文件之外，在MATLAB中可以使用C语言来编写读取MAT文件的MEX文件或者EXE独立应用程序。关于这方面的具体内容，请读者查看对应的MATLAB帮助文件。

14.2.2 使用FORTRAN语言创建MAT文件

上面介绍了如何使用C语言来创建MAT文件，本小节将利用一个简单例子来介绍如何使用FORTRAN语言来创建MAT文件。

例 14.5 使用C语言编写创建MAT文件的程序代码。

step 1 打开在系统中安装的FORTRAN语言开发工具（在本实例中使用的是Compaq Visual Fortran），然后在开发工具中输入下面的程序代码

```
C     matdemo1.f
C     This is a simple program that illustrates how to call the
C     MATLAB MAT-file functions from a Fortran program.  This
C     demonstration focuses on writing MAT-files.
C
C     matdemo1 - Create a new MAT-file from scratch.
      program matdemo1
      integer matOpen, matClose
      integer matGetVariable, matPutVariable
      integer matPutVariableAsGlobal, matDeleteVariable
      integer mxCreateDoubleMatrix, mxCreateString
      integer mxIsFromGlobalWS, mxGetPr
      integer mp, pa1, pa2, pa3, pa0, status
      double precision dat(9)
      data dat / 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0 /
C     Open MAT-file for writing.
      write(6,*) 'Creating MAT-file matdemo.mat ...'
      mp = matOpen('matdemo.mat', 'w')
      if (mp .eq. 0) then
         write(6,*) 'Can''t open ''matdemo.mat'' for writing.'
         write(6,*) '(Do you have write permission in this
                     directory?)'
         stop
      end if
C     创建变量
      pa0 = mxCreateDoubleMatrix(3,3,0)
      call mxCopyReal8ToPtr(dat, mxGetPr(pa0), 9)
      pa1 = mxCreateDoubleMatrix(3,3,0)
      pa2 = mxCreateString('MATLAB: The language of computing')
      pa3 = mxCreateString('MATLAB: The language of computing')
      status = matPutVariableAsGlobal(mp, 'NumericGlobal', pa0)
      if (status .ne. 0) then
         write(6,*) 'matPutVariableAsGlobal ''Numeric Global''
                    failed'
         stop
      end if
      status = matPutVariable(mp, 'Numeric', pa1)
```

```
      if (status .ne. 0) then
         write(6,*) 'matPutVariable ''Numeric'' failed'
         stop
      end if
      status = matPutVariable(mp, 'String', pa2)
      if (status .ne. 0) then
         write(6,*) 'matPutVariable ''String'' failed'
         stop
      end if
      status = matPutVariable(mp, 'String2', pa3)
      if (status .ne. 0) then
         write(6,*) 'matPutVariable ''String2'' failed'
         stop
      end if
C
      call mxCopyReal8ToPtr(dat, mxGetPr(pa1), 9)
      status = matPutVariable(mp, 'Numeric', pa1)
      if (status .ne. 0) then
         write(6,*) 'matPutVariable ''Numeric'' failed 2nd time'
         stop
      end if
C     从MAT文件中删除String2 变量
      status = matDeleteVariable(mp, 'String2')
      if (status .ne. 0) then
         write(6,*) 'matDeleteVariable ''String2'' failed'
         stop
      end if
C     重新阅读MAT文件
      status = matClose(mp)
      if (status .ne. 0) then
         write(6,*) 'Error closing MAT-file'
         stop
      end if
      mp = matOpen('matdemo.mat', 'r')
      if (mp .eq. 0) then
         write(6,*) 'Can''t open ''matdemo.mat'' for reading.'
         stop
      end if
      pa0 = matGetVariable(mp, 'NumericGlobal')
      if (mxIsFromGlobalWS(pa0) .eq. 0) then
         write(6,*) 'Invalid non-global matrix written to MAT-file'
         stop
      end if
      pa1 = matGetVariable(mp, 'Numeric')
      if (mxIsNumeric(pa1) .eq. 0) then
         write(6,*) 'Invalid non-numeric matrix written to
                    MAT-file'
         stop
      end if
      pa2 = matGetVariable(mp, 'String')
      if (mxIsString(pa2) .eq. 0) then
         write(6,*) 'Invalid non-string matrix written to MAT-file'
         stop
      end if
      pa3 = matGetVariable(mp, 'String2')
      if (pa3 .ne. 0) then
```

```
            write(6,*) 'String2 not deleted from MAT-file'
            stop
         end if
C        Clean up memory.
         call mxDestroyArray(pa0)
         call mxDestroyArray(pa1)
         call mxDestroyArray(pa2)
         call mxDestroyArray(pa3)
         status = matClose(mp)
         if (status .ne. 0) then
            write(6,*) 'Error closing MAT-file'
            stop
         end if
         write(6,*) 'Done creating MAT-file'
         stop
         end
```

在输入上面的程序代码后，将该程序代码保存为 matdemo1.f，然后将该文件保存到用户使用 MATLAB 的目标路径中。

step 2 编译上面的程序代码。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>>mex matdemo1.f
```

上面的程序代码将会产生一个 MAT 文件 matdemo.mat，用户可以向 MATLAB 中加载该数据文件。向 MATLAB 的命令窗口中输入"matdemo1"，得到的结果如下：

```
Creating MAT-file matdemo.mat ...
Done creating MAT-file
```

step 3 查看加载的结果。在 MATLAB 的命令窗口中输入"whos -file matdemo.mat"，得到的结果如下：

```
Name           Size          Bytes  Class
 Numeric        3x3              72  double array
 String         1x33             66  char array

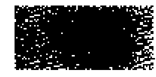
Grand total is 42 elements using 138 bytes
```

step 4 查看变量的结果。在命令窗口中输入变量名称，就可以查看具体的参数数值如下：

```
Numeric =
     1     2     3
     4     5     6
     7     8     9
String =
MATLAB: The language of computing
```



除了可以编写创建MEX文件之外，在MATLAB中可以使用FORTRAN语言来编写读取MAT文件的MEX文件或者EXE独立应用程序。关于这方面的具体内容，请查看对应的帮助文件。



MATLAB 引擎技术

前面花费了一定的篇幅来介绍MEX文件，本节将会介绍另外一种和该文件思想完全相反的内容——MATLAB 引擎技术，也就是在其他应用程序中调用MATLAB的程序，例如调用MATLAB的Math库，进行数值计算等。

引擎技术概念

MATLAB 中拥有一个引擎库。在该引擎库中汇集了多种函数，用户可以在自行编写的程序代码中引用这些函数，实现对MATLAB的调用。也就是说，用户可以自行编写界面运行在前台，而MATLAB作为计算引擎后台。引擎函数本身是使用C语言或者FORTRAN编写的，在Windows平台中，它和MATLAB之间的通信是通过ActiveX实现的。MATLAB 引擎可以运用在下面的场合中：

- MATLAB在由其他语言编写的应用程序中被当作数学库程序调用，这样就可以在其他应用程序中利用 MATLAB 命令简单、计算可靠的优点；
- MATLAB在专门系统中当作计算引擎使用时，前台是其他应用程序语言所编写的GUI图形接口，后台由 MATLAB 来进行计算，这样就可以节省用户的开发时间。

引擎技术应用

在本小节中将利用一个简单的实例来介绍如何使用 C 语言编写程序代码，在该程序代码中调用 MATLAB 计算引擎。

例 14.6 使用 C 语言编写引擎应用的实例。

step 1 打开用户系统中安装的 C 语言开发工具，然后在开发工具中输入下面的程序代码：

```
/*
 *  engwindemo.c
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "engine.h"
#define BUFSIZE 256
static double Areal[ 6] = { 1, 2, 3, 4, 5, 6 };
int PASCAL WinMain (HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR     lpszCmdLine,
                    int       nCmdShow)
{
% 定义 ep 为 MATLAB 引擎的指针
   Engine *ep;
% 定义三个空的结构体
   mxArray *T = NULL, *a = NULL, *d = NULL;
% 定义容量为 257 的缓冲区
   char buffer[ BUFSIZE+1] ;
% 定义双精度变量的指针
   double *Dreal, *Dimag;
% 定义双精度变量
   double time[ 10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
```

```
/*
 * 启动MATLAB引擎，如果出现错误则退出程序
 */
if (!(ep = engOpen(NULL))) {
      MessageBox ((HWND)NULL, (LPSTR)"Can't start MATLAB engine",
            (LPSTR) "Engwindemo.c", MB_OK);
      exit(-1);
}
/*
 * 创建变量
T = mxCreateDoubleMatrix(1, 10, mxREAL);
memcpy((char *) mxGetPr(T), (char *) time, 10*sizeof(double));
/*
 * 将变量T传递到MATLAB的工作空间中
engPutVariable(ep, "T", T);
/*
 * 根据公式 distance = (1/2)g.*t.^2计算变量数值
      engEvalString(ep, "D = .5.*(-9.8).*T.^2;");
/ * 绘制计算结果
engEvalString(ep, "plot(T,D);");
engEvalString(ep, "title('Position vs. Time for a falling object');");
engEvalString(ep, "xlabel('Time (seconds)');");
engEvalString(ep, "ylabel('Position (meters)');");
 /* 计算特征值数值
       engEvalString(ep, "d = eig(A*A')");
 /*
 * 获取MATLAB的输出变量
 buffer[BUFSIZE] = '\0';
 engOutputBuffer(ep, buffer, BUFSIZE);
 /*
 * 返回计算数值到缓冲区中
 engEvalString(ep, "whos");
 MessageBox ((HWND)NULL, (LPSTR)buffer, (LPSTR) "MATLAB - whos",
MB_OK);
 /*
 * 计算特征值数据矩阵
 d = engGetVariable(ep, "d");
%关闭ep所指向的引擎
 engClose(ep);
 if (d == NULL) {
          MessageBox ((HWND)NULL, (LPSTR)"Get Array Failed", (LPSTR)
"Engwindemo.c", MB_OK);
      }
else {
      Dreal = mxGetPr(d);
      Dimag = mxGetPi(d);
      if (Dimag)
          sprintf(buffer,"Eigenval 2: %g+%gi",Dreal[1],Dimag[1]);
      else
          sprintf(buffer,"Eigenval 2: %g",Dreal[1]);
      MessageBox ((HWND)NULL, (LPSTR)buffer, (LPSTR)"Engwindemo.c",
MB_OK);
     mxDestroyArray(d);
}
/* 释放所有的内存空间
mxDestroyArray(T);
```

```
    mxDestroyArray(a);
    return(0);
}
```

完成上面的程序代码后，将程序代码保存为“engwindemo.c”文件，然后将该文件保存到用户使用 MATLAB 的目录路径中。

step 2 编译上面的程序代码，在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> mex -f D:\SOFTWARE\MATLAB7.0\bin\win32\mexopts\lccengmatopts.bat ...
   engwindemo.c
```

step 3 运行编译文件。输入代码后，按“Enter”键，在对应的目录路径中会创建 exe 文件，可以双击该文件或者在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> !engwindemo&
```

step 4 查看程序代码的结果。输入代码后，按“Enter”键，得到的图形如图 14.2 所示。

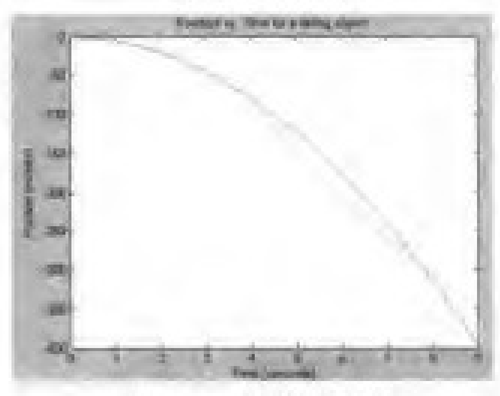


图 14.2　计算结果图形

在显示计算结果图形的同时，MATLAB 还将启动一个进程，该进程仅仅包含 MATLAB 的命令窗口，如图 14.3 所示。

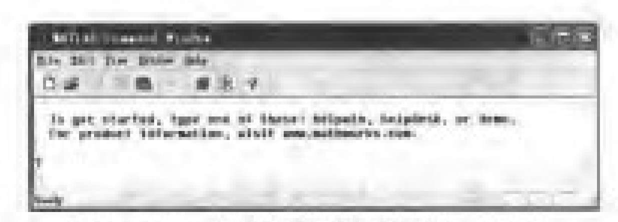


图 14.3　在运行程序时显示的对话框

step 5 查看程序变量的结果。在该命令窗口中查看程序运行的所有的变量结果：

```
?whos
  Name      Size                    Bytes  Class
  A         3x2                        48  double array
  D         1x10                       80  double array
  T         1x10                       80  double array
  d         3x1                        24  double array

Grand total is 29 elements using 232 bytes
?A
```

```
A =
     1     4
     2     5
     3     6
?D'
ans =
         0
   -4.9000
  -19.6000
  -44.1000
  -78.4000
 -122.5000
 -176.4000
 -240.1000
 -313.6000
 -396.9000
?T'
ans =
     0
     1
     2
     3
     4
     5
     6
     7
     8
     9
?d'
ans =
   -0.0000    0.5973   90.4027
?
```

最后，根据应用程序代码，该程序还会显示程序变量信息的对话框，如图 14.4 所示。

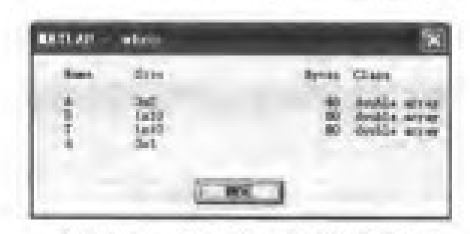


图 14.4　程序变量信息对话框

在本小节的最后，笔者将借上面的例子总结使用 C 语言编写引擎文件的注意事项。

- 首先，在上面的程序代码中，在代码的最开始包含了程序所需的头文件。在所有的 C 语言引擎应用程序中，都必须包含 engine.h 头文件，因为该头文件声明了所有 eng 函数的原型。
- 在程序代码中，一般需要首先定义 Engine 类型的指针。该指针类似于打开文件时的文件指针，相当于计算引擎的接口句柄。有了这个指针就可以在 C 语言中执行 MATLAB 的各种命令。
- 在程序代码的最后，一般需要关闭计算引擎，这个工作需要函数 engClose 来完成，通过该命令可以关闭指针来释放内存。
- MATLAB 的计算引擎应用程序的基础流程是打开计算引擎、设置数据、执行 MATLAB 命令、获取计算结果，最后关闭计算引擎。



对于上面的程序流程，MATLAB分别提供了eng类函数完成各种程序功能。关于这些eng类函数的具体功能，请自行查阅对应的帮助文件。

14.4 Java 接口

Java 语言是当前比较流行的面向对象的高级编程语言，能够完成各种类型的应用程序开发。MATLAB和Java之间的关系是非常密切的，从5.x版本开始，MATLAB中就包含Java虚拟机，在MATLAB中可以直接调用Java应用程序。Java可以填补MATLAB在功能上的一些空白，同时由于Java本身的优势，可以通过Java语言获得大量来自互联网或者数据库中的数据，而MATLAB的优势则是对数据进行分析、科学计算等，充分发挥各自的优势，可以极大地提高工作效率。

在本节中，将首先介绍MATLAB中的Java接口语言基础内容，然后以一个综合实例来介绍如何使用Java语言编写综合应用实例。

14.4.1 Java 接口语言基础

在MATLAB中使用Java语言之前，首先有必要了解当前MATLAB所使用的Java虚拟机的版本，用户可以使用 "version -java" 命令得到版本信息，如下：

```
>> version -java
ans =
Java 1.4.2 with Sun Microsystems Inc. Java HotSpot(TM) Client VM
     (mixed mode)
```

根据上面的版本信息，用户选择合适的JDK版本，避免出现不兼容的情况。

Java是一种面向对象的高级程序语言。在这种程序语言中，类和对象是最基础的概念。如果用户需要创建对象，首先必须有对应的类存在。在MATLAB中，用户可以使用三种Java类，即Java内置的函数类、第三方定义类和用户自定义类。

在MATLAB中，提供javaclasspath来加载和显示Java内建类和由Mathworks公司提供的第三方定义类，用户可以使用该命令查看已经存在的Java类，结果如下：

```
>> javaclasspath
      STATIC JAVA PATH

  D:\SoftWare\MATLAB7.0\java\patch
  D:\SoftWare\MATLAB7.0\java\jar\util.jar
  D:\SoftWare\MATLAB7.0\java\jar\widgets.jar
  D:\SoftWare\MATLAB7.0\java\jar\beans.jar
  D:\SoftWare\MATLAB7.0\java\jar\hg.jar
  D:\SoftWare\MATLAB7.0\java\jar\ice.jar
  D:\SoftWare\MATLAB7.0\java\jar\ide.jar
  D:\SoftWare\MATLAB7.0\java\jar\jmi.jar
  D:\SoftWare\MATLAB7.0\java\jar\mde.jar
  D:\SoftWare\MATLAB7.0\java\jar\mlservices.jar
  D:\SoftWare\MATLAB7.0\java\jar\mlwidgets.jar
  D:\SoftWare\MATLAB7.0\java\jar\mwswing.jar
.......................................................................//省略了部分数据
  D:\SoftWare\MATLAB7.0\java\jarext\J2PrinterWorks.jar
```

```
D:\SoftWare\MATLAB7.0\java\jarext\jaccess-1_4.jar
D:\SoftWare\MATLAB7.0\java\jarext\junit.jar
D:\SoftWare\MATLAB7.0\java\jarext\mwucarunits.jar
D:\SoftWare\MATLAB7.0\java\jarext\saxon.jar
D:\SoftWare\MATLAB7.0\java\jarext\vb20.jar
D:\SoftWare\MATLAB7.0\java\jarext\wsdl4j.jar
D:\SoftWare\MATLAB7.0\java\jarext\xalan.jar
D:\SoftWare\MATLAB7.0\java\jarext\xercesImpl.jar
D:\SoftWare\MATLAB7.0\java\jarext\xml-apis.jar

        DYNAMIC JAVA PATH
<empty>
```

从上面的结果中可以看出，在默认情况下将分为静态和动态Java路径，其中静态路径主要用来保存稳定、静态的Java类，而对于需要编辑的Java类，则建议保存在动态路径中。在默认情况下，classpath.txt文件会保存在toolbox\local路径下，具体信息如下：

```
>> which classpath.txt
D:\SoftWare\MATLAB7.0\toolbox\local\classpath.txt
```

在默认情况下，MATLAB本身会自动加载Java的内置函数类。为了查看当前使用的MATLAB中加载的所有函数类名，可以使用inmem命令查看所有的类名：

```
>> [m,x,j] = inmem
m =
    'matlabrc'
    'pathdef'
    'userpath'
    'ispc'
    'filesep'
    'pwd'
    'usejava'
    'hgrc'
    'opaque.char'
    'colordef'
    'whitebg'
    'jet'
    'initprefs'
    'findallwinclasses'
    'initdesktoputils'
    'path'
    'mdbstatus'
    'workspacefunc'
    'num2str'
    'mat2str'
    'int2str'
    'strvcat'
    'javaclasspath'
    'pathsep'
    'iscellstr'
x =
    'cellfun'
j =
    'java.util.Locale'
    'GObject'
```

```
'schema.class'
'figure'
'schema.method'
'java.lang.String'
'java.lang.CharSequence'
'com.mathworks.jmi.ClassLoaderManager'
```



在上面的程序代码中，m数组表示的内容是系统加载的M文件，x表示系统加载的MEx文件，j表示系统加载的Java函数类。

在MATLAB中，可以直接使用Java类定义和javaObject函数来创建Java对象，这两种使用方法都比较简单，下面使用实例来简要说明。

例 14.7 在MATLAB中创建Java对象。

step 1 创建Java对象。在MATLAB的命令窗口中输入下面的程序代码：

```
>>  url = java.net.URL(...
     'http://archive.ncsa.uiuc.edu/demoweb/');
frame = java.awt.Frame('Frame Java B');
class  = 'java.lang.String';
text = 'hello';
strObj = javaObject(class, text);
origFrame = java.awt.Frame;
setSize(origFrame, 800, 400);
newFrameRef = origFrame;
setSize(newFrameRef, 1000, 800);
getSize(origFrame);
```

step 2 查看程序代码的结果。在命令窗口中输入"whos"，然后按"Enter"键，得到的结果如下：

```
>> whos
  Name              Size                    Bytes  Class
  ans               1x1                            java.awt.Dimension
  class             1x16                       32  char array
  frame             1x1                            java.awt.Frame
  newFrameRef       1x1                            java.awt.Frame
  origFrame         1x1                            java.awt.Frame
  strObj            1x1                            java.lang.String
  text              1x5                        10  char array
  url               1x1                            java.net.URL

Grand total is 27 elements using 42 bytes
```

step 3 查看Java对象。在MATLAB的命令窗口中查看具体的变量信息如下：

```
>> url
url =
http://archive.ncsa.uiuc.edu/demoweb/
>> frame
frame =
java.awt.Frame[ frame1,0,0,0x0,invalid,hidden,layout=java.awt.
BorderLayout,title=Frame Java B,resizable,normal]
```

```
>> strObj
 strObj =
hello
>> newFrameRef
newFrameRef =
java:awt.Frame[ frame2,0,0,1000x800,invalid,hidden,layout=java.awt.
BorderLayout,title=,resizable,normal]
>> getSize(origFrame)
ans =
java.awt.Dimension[ width=1000,height=800]
```



从上面的程序代码中，分别使用 Java 内置函数和 javaObject 函数来创建 Java 对象，读者可以从上面的例子中了解到 MATLAB 中创建 Java 对象的方法。

例 14.8 在 MATLAB 中对 Java 对象进行操作。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> point1 = java.awt.Point(24,127);
point2 = java.awt.Point(114,29);
>> point=cat(1, point1, point2);
>> byte = java.lang.Byte(127);
integer = java.lang.Integer(52);
double = java.lang.Double(7.8);
>> number=[ byte; integer; double];
```

step 2 查看变量的结果。在命令窗口输入变量名称，得到的结果如下：

```
>> point
point =
java.awt.Point[ ]:
    [ java.awt.Point]
    [ java.awt.Point]
>> number
number =
 java.lang.Number[ ]:
    [    127]
    [     52]
    [ 7.8000]
```



在上面的程序代码中，首先直接使用 Java 类定义创建了 Java 对象，然后分别根据变量类型不同将两组不同的变量进行连接操作。

例 14.9 在 MATLAB 中设置 Java 对象的属性。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> frame=java.awt.Frame('A')
frame =
java.awt.Frame[ frame2,0,0,0x0,invalid,hidden,layout=java.awt.
BorderLayout,title=A,resizable,normal]
>> setTitle(frame, 'Sample Frame')
>> frame
```

```
frame =
java.awt.Frame[frame2,0,0,0x0,invalid,hidden,layout=java.awt.
BorderLayout,title=Sample Frame,resizable,normal]
>> title = getTitle(frame)
title =
Sample Frame
```

在上面的程序代码中，使用了setTitle命令设置了frame对象的标题属性，然后使用getTitle命令来获取该对象的属性。

step 2 使用 javaMethod 函数设置 Java 对象的属性。在命令窗口中输入下面的程序代码：

```
>> gAddress = java.lang.String('Four score and seven years ago');
str = java.lang.String('Four score');
javaMethod('startsWith', gAddress, str)
ans =

     1
```



在上面的程序代码中，使用的 Java 方法是 startsWith，检测 gAddress 字符串对象是否以 "str" 字符串对象作为开头，得到的结果是 1，表明检测结果正确。也就是说，gAddress 字符串对象的确以 "str" 字符串对象作为开头。

例 14.10 在 MATLAB 中创建 Java 类型的数据。

step 1 创建 Java 数据类型的数组。在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> origArray = javaArray('java.lang.Double', 3, 4);
for m = 1:3
   for n = 1:4
      origArray(m,n) = java.lang.Double((m * 10) + n);
   end
end
>> origArray
```

step 2 查看程序代码的结果。输入代码后，按 "Enter" 键，得到如下的结果：

```
origArray =
java.lang.Double[][]:
    [11]     [12]     [13]     [14]
    [21]     [22]     [23]     [24]
    [31]     [32]     [33]     [34]
```



在上面的程序中，首先使用 javaArray 命令创建了一个 3 行 4 列的 Java 数值数组，而且该数组中的数值类型为 java.lang.Double，然后使用循环语句来添加每个数组的数值，得到最后的结果。

例 14.11 在 MATLAB 中，将 Java 类型的数据转换为 MATLAB 中的元胞数组。

step 1 在 MATLAB 的命令窗口中输入下面的程序代码：

```
>> import java.lang.* java.awt.*;
% Create a Java array of double
```

```
dblArray = javaArray('java.lang.Double', 1, 10);
for m = 1:10
   dblArray(1, m) = Double(m * 7);
end
% Create a Java array of points
ptArray = javaArray('java.awt.Point', 3);
ptArray(1) = Point(7.1, 22);
ptArray(2) = Point(5.2, 35);
ptArray(3) = Point(3.1, 49);
% Create a Java array of strings
strArray = javaArray('java.lang.String', 2, 2);
strArray(1,1) = String('one');    strArray(1,2) = String('two');
strArray(2,1) = String('three');  strArray(2,2) = String('four');
% Convert each to cell arrays
cellArray = {cell(dblArray), cell(ptArray), cell(strArray)}
```

step 2 查看程序代码的结果。输入代码后，按"Enter"键，得到如下的结果

```
cellArray =
    {1x10 cell}    {3x1 cell}    {2x2 cell}
>> cellArray{1,1}
ans =
    [7]    [14]    [21]    [28]    [35]    [42]    [49]    [56]    [63]
[70]
>> cellArray{1,2}
ans =
    [1x1 java.awt.Point]
    [1x1 java.awt.Point]
    [1x1 java.awt.Point]
>> cellArray{1,3}
ans =
     'one'      'two'
     'three'    'four'
```



在上面的程序代码中，依次创建了Java类型的双精度、points和字符串数据类型，然后使用cell命令将其转换为MATLAB中的元胞数组。

14.4.2 Java 接口应用

在本小节中，将以一个比较综合的案例来说明如何在MATLAB中编写Java接口程序代码。该程序代码的主要功能是实现用户电话号码的交互操作，实现数据的添加、删除、显示等综合功能，下面分步骤详细介绍。

例 14.12 在MATLAB中编写用户电话号码本的Java接口程序代码。

step 1 打开M语言编辑器，输入下面的程序代码：

```
function phonebook(varargin)
pbname = 'myphone';
%处理原始数据文件的名称和路径
if ispc
   datadir = char(java.lang.System.getProperty('user.dir'));
else
   datadir = getenv('HOME');
```

```
end;
pbname = fullfile(datadir, pbname)
%如果不存在文件，创建该文件
if ~exist(pbname)
   disp(sprintf('Data file %s does not exist.', pbname));
   r = input('Create a new phone book (y/n)?','s');
   if r == 'y',
      try
         FOS = java.io.FileOutputStream(pbname);
         FOS.close
      catch
         error(sprintf('Failed to create %s', pbname));
      end;
   else
      return;
   end;
end;
pb_htable = java.util.Properties;
try
   FIS = java.io.FileInputStream(pbname);
catch
   error(sprintf('Failed to open %s for reading.', pbname));
end;
pb_htable.load(FIS);
FIS.close;

while 1
%显示用户选择的选项
   disp ' '
   disp ' Phonebook Menu:'
   disp ' '
   disp ' 1. Look up a phone number'
   disp ' 2. Add an entry to the phone book'
   disp ' 3. Remove an entry from the phone book'
   disp ' 4. Change the contents of an entry in the phone book'
   disp ' 5. Display entire contents of the phone book'
   disp ' 6. Exit this program'
   disp ' '
%获取用户选择的选项
   s = input('Please type the number for a menu selection: ','s');
switch s
   case '1',
      name = input('Enter the name to look up: ','s');
      if isempty(name)
         disp 'No name entered'
      else
%调用查看函数
         pb_lookup(pb_htable, name);
      end;
   case '2',
%调用添加函数
      pb_add(pb_htable);
   case '3',
      name=input('Enter the name of the entry to remove: ', 's');
      if isempty(name)
         disp 'No name entered'
```

```
        else
% 调用删除函数
            pb_remove(pb_htable, name);
        end;
    case '4',
        name=input('Enter the name of the entry to change: ', 's');
        if isempty(name)
            disp 'No name entered'
        else
% 调用修改函数
            pb_change(pb_htable, name);
        end;
    case '5',
% 调用显示列表函数
        pb_listall(pb_htable);
    case '6',
        try
            FOS = java.io.FileOutputStream(pbname);
        catch
            error(sprintf('Failed to open %s for writing.',...
                          pbname));
        end;
        pb_htable.save(FOS,'Data file for phonebook program');
        FOS.close;
        return;
    otherwise
        disp 'That selection is not on the menu.'
    end;
end;
```

上面的程序代码是该程序代码的主函数，在程序代码的开头首先处理电话号码文件的路径，如果用户在运行程序代码之前已经创建电话号码文件，则返回该文件的路径全称；如果没有创建电话号码文件，则重新创建java.io.FileOutputStream对象来添加电话号码数据。当程序代码创建电话号码对象后，则提供用户选择对应的操作，然后主函数将需要调用对应的子函数完成对应的操作。

step 2 添加所有的子函数程序代码。在M语言编辑器中输入下面的程序代码：

```
function pb_lookup(pb_htable,name)
entry = pb_htable.get(pb_keyfilter(name));
if isempty(entry),
    disp(sprintf('The name %s is not in the phone book',name));
else
    pb_display(entry);
end
% 添加号码的子程序
function pb_add(pb_htable)
disp 'Type the name for the new entry, followed by Enter.'
disp 'Then, type the phone number(s), one per line.'
disp 'To complete the entry, type an extra Enter.'
name = input(':: ','s');
entry=[ name '^'];
while 1
    line = input(':: ','s');
    if isempty(line)
```

```
        break;
    else
        entry=[ entry line '^'];
    end;
end;
if strcmp(entry, '^')
    disp 'No name entered'
    return;
end;
%添加对应的电话号码
pb_htable.put(pb_keyfilter(name),entry);
disp ' '
disp(sprintf('%s has been added to the phone book.', name));
%删除号码的子程序
function pb_remove(pb_htable,name)
if ~pb_htable.containsKey(pb_keyfilter(name))
    disp(sprintf('The name %s is not in the phone book',name))
    return
end;
r = input(sprintf('Remove entry %s (y/n)? ',name), 's');
if r == 'y'
%删除选中的电话号码
    pb_htable.remove(pb_keyfilter(name));
    disp(sprintf('%s has been removed from the phone book',name))
else
    disp(sprintf('%s has not been removed',name))
end;
%修改号码的子程序
function pb_change(pb_htable,name)
entry = pb_htable.get(pb_keyfilter(name));
if isempty(entry)
    disp(sprintf('The name %s is not in the phone book', name));
    return;
else
    pb_display(entry);
    r = input('Replace phone numbers in this entry (y/n)? ','s');
    if r ~= 'y'
        return;
    end;
end;
disp 'Type in the new phone number(s), one per line.'
disp 'To complete the entry, type an extra Enter.'
disp(sprintf(':: %s', name));
entry=[ name '^'];
while 1
    line = input(':: ','s');
    if isempty(line)
        break;
    else
        entry=[ entry line '^'];
    end;
end;
%完成电话号码的修改
pb_htable.put(pb_keyfilter(name),entry);
disp ' '
disp(sprintf('The entry for %s has been changed', name));
```

```
% 显示电话号码列表的子程序
function pb_listall(pb_htable)
enum = pb_htable.propertyNames;
while enum.hasMoreElements
   key = enum.nextElement;
% 调用pb_display函数
   pb_display(pb_htable.get(key));
end;
% 显示号码的子程序
function pb_display(entry)
disp ' '
disp '-------------------------'
[ t,r] = strtok(entry,'^');
while ~isempty(t)
   disp(sprintf(' %s',t));
   [ t,r] = strtok(r,'^');
end;
disp '-------------------------'
function out = pb_keyfilter(key)
if ~isempty(findstr(key,' '))
   out = strrep(key,' ','_');
else
   out = strrep(key,'_',' ');
end;
```

完成上面的程序代码后，将所有的程序代码保存为"phonebook.m"文件，然后将其保存到用户所使用的MATLAB路径中。

step 3 运行程序代码。在MATLAB的命令窗口中输入"phonebook"，得到如下的结果：

```
Phonebook Menu:

     1. Look up a phone number
     2. Add an entry to the phone book
     3. Remove an entry from the phone book
     4. Change the contents of an entry in the phone book
     5. Display entire contents of the phone book
     6. Exit this program

Please type the number for a menu selection: 5

-------------------------
 Sylvia Woodland
 (508) 111-3456
-------------------------
-------------------------
 Russell Reddy
 (617) 999-8765
-------------------------
```

step 4 添加新的数据。在上面的程序代码中，用户查看了原始的数据文件。在后面的步骤中，可以在该文件中添加新的数据，具体的信息如下：

```
Phonebook Menu:

     1. Look up a phone number
     2. Add an entry to the phone book
```

```
     3. Remove an entry from the phone book
     4. Change the contents of an entry in the phone book
     5. Display entire contents of the phone book
     6. Exit this program

Please type the number for a menu selection: 2

Type the name for the new entry, followed by Enter.
Then, type the phone number(s), one per line.
To complete the entry, type an extra Enter.
:: BriteLites Books
:: (781) 777-6868
::

BriteLites Books has been added to the phone book.

Phonebook Menu:

     1. Look up a phone number
     2. Add an entry to the phone book
     3. Remove an entry from the phone book
     4. Change the contents of an entry in the phone book
     5. Display entire contents of the phone book
     6. Exit this program

Please type the number for a menu selection: 5

-------------------------
 BriteLites Books
 (781) 777-6868
-------------------------

-------------------------
 Sylvia Woodland
 (508) 111-3456
-------------------------

-------------------------
 Russell Reddy
 (617) 999-8765
-------------------------
```



用户还可以检测该程序代码的其他功能，在本章中，限于篇幅，在这里就不一一检测具体功能了，请用户自行尝试。

14.5 小结

在本章中，主要向读者介绍了在MATLAB中如何使用C或者FORTRAN语言创建MEX文件和MAT文件，然后介绍了MATLAB的引擎技术和Java接口的内容。这些内容是MATLAB程序接口的重要内容，希望用户仔细分析。

◆ ◆ ◆

反侵权盗版声明